UNIVERSITÉ DE LIMOGES

ÉCOLE DOCTORALE Sciences et Ingénierie pour l'Information

FACULTÉ des Sciences et Techniques

Institut de recherche XLIM

Thèse N° 52-2010

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Spécialité : Informatique et Applications

présentée et soutenue par

MICKAËL POUCHOL

le 05 Novembre 2010

STRUCTURES HIÉRARCHIQUES POUR LA SIMULATION DE FLUIDES

Thèse dirigée par Djamchid Ghazanfarpour
co-encadrée par Benoît Crespin

JURY :

rapporteurs		
Richard Egli	Professeur	Université de Sherbrooke
Christophe Schlick	Professeur	Université Bordeaux 2
examinateurs		
Sébastien Thon	Maître de conférences	Université de Provence
Djamchid Ghazanfarpour	Professeur	Université de Limoges
Benoît Crespin	Maître de conférences	Université de Limoges

Remerciements

En tout premier lieu, je tiens à exprimer ma reconnaissance envers mes encadrants, M. Djamchid Ghazanfarpour ainsi que M. Benoît Crespin à la fois pour leurs conseils et pour le temps qu'ils m'ont consacré.

Je souhaite également exprimer ma gratitude à l'égard de M. Richard Egli et de M. Christophe Schlick pour avoir accepté d'être rapporteurs de mes travaux de thèse, ainsi qu'à l'égard de M. Sébastien Thon qui a accepté de faire partie du jury.

Je tiens également à remercier mes collègues : Ahmadou, Alexandre, Agnès, Boussad, Céline, Cyril, Emmanuelle, Nadir, Oana, Richard à qui j'adresse mes profonds remerciements pour leur soutien. Plus particulièrement, je les remercie pour les bons moments que nous avons partagés durant ces trois années.

Enfin je tiens à remercier ma famille qui m'a toujours encouragé et soutenu dans cette voie et m'a permis d'arriver jusqu'ici.

Résumé

La simulation de l'écoulement des fluides comme l'eau ou la fumée est un des phénomènes naturels les plus traités par la communauté de l'informatique graphique. Parmi ces méthodes, les méthodes lagrangiennes, représentant le fluide à l'aide de particules, sont souvent préférées car elles permettent d'utiliser moins de ressources, elles restent cependant coûteuses en temps de calculs et nécessitent l'usage de structures de données adaptées lors de différentes phases. Nous proposons donc d'utiliser une structure de hachage hiérarchique afin de traiter la phase de détection de collisions avec des objets répartis de manière irrégulière et de tailles très différentes. L'utilisation de plusieurs niveaux de hachages permet de bénéficier de la cohérence spatiale des objets et de réduire significativement les collisions de hachage. Les méthodes adaptatives permettent d'allouer efficacement les ressources de calculs dans les zones importantes du fluide en faisant varier la taille des particules de la simulation. Dans cette situation, une structure de hachage ou de grille hiérarchique permet de traiter efficacement la phase de recherche de voisinage en insérant chaque particule dans le niveau de la grille approprié mais aussi de déterminer les zones de fusion ou de subdivision grâce à des critères locaux. Enfin la phase de visualisation du fluide représente un défi car le résultat est directement sanctionné par le spectateur. Dans le cas des méthodes lagrangiennes, la méthode des blobs traditionnellement utilisée ne permet pas d'obtenir une surface de qualité. Nous utilisons pour cette tâche la méthode des surfaces variationnelles.

Abstract

Fluid flow simulation such as water or smoke is one of the most covered natural phenomena in the computer graphics community. Among these methods, lagrangian ones which divide the fluid into a set of particles, are usually favored for their efficiency, however they require suitable data structures in several steps to reduce their high computation times. We suggest to use a hierarchical hash table to handle the collision detection step with irregularly distributed and objects with variable sizes. The use of several hash levels allows to benefit from objects spatial coherence and to drastically reduce hash collisions. Adaptive methods allow to efficiently allocate computational resources to significant fluid regions by varying particles sizes during the simulation. In this case a hierarchical grid or hash table structure allows to efficiently handle nearest neighbors search by inserting each particle in the appropriate grid level, and also allows to define appropriate merging or splitting regions with local criteria. Finally the visualization step is challenging because the end-user will appreciate the final result directly. For lagrangian methods, the most common way to deal with this problem is to use blob-based methods that do not yield satisfying results ; we use the variational surface method for this task.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Introduction

La reproduction des phénomènes naturels représente une étape nécessaire dans la quête de photo-réalisme poursuivie par les créateurs de jeux vidéo et les réalisateurs de films en images de synthèses. Parmi ces phénomènes, la simulation de l'écoulement des fluides comme l'eau ou la fumée est probablement le plus traité par les chercheurs de la communauté de l'informatique graphique ainsi que de la physique numérique. En effet il s'agit d'un phénomène complexe dont la simulation reste difficile et coûteuse en ressources de calcul. Il est d'autant plus difficile de visualiser efficacement ces phénomènes qu'ils sont visibles en permanence dans la vie quotidienne, il est donc peu probable d'arriver à tromper un spectateur avec une représentation simplifiée. Ces problèmes sont encore plus présents dans le cas d'applications interactives où les ressources sont très limitées. Dans ce contexte, les méthodes lagrangiennes sont particulièrement appréciées pour leur simplicité et efficacité. Ces méthodes représentent un fluide par un ensemble de particules, assimilables à des atomes de fluide, dont le mouvement et les interactions déterminent son écoulement global. Elles permettent aussi d'allouer naturellement les ressources de calcul aux régions pertinentes, c'est à dire où se produit l'écoulement, évitant ainsi de gaspiller des ressources de calcul pour les espaces vides du fluide. Les méthodes lagrangiennes sont par exemple utilisées dans l'industrie des effets spéciaux avec le logiciel RealFlow [1]. L'utilisation des méthodes lagrangiennes dans ce logiciel permet de simuler des effets difficilement réalisables avec d'autres méthodes comme des éclaboussures et des jets d'eau.

Le coût élevé des méthodes de simulation de fluide a fortement inspiré les chercheurs afin de trouver des solutions permettant une simulation plus efficace sur des machines conventionnelles. Cependant l'écoulement d'un fluide reste un phénomène difficile à simuler de manière réaliste en temps réel. Les jeux vidéo par exemple, utilisent généralement des trucages assez grossiers qui peinent à convaincre et se contentent le plus souvent de simuler des phénomènes peu turbulents se produisant à la surface de flaques, d'étangs ou de l'océan. Les phénomènes tels que les éclaboussures sont généralement produits avec des aplats de texture (Figure 1). L'écoulement du fluide en lui même n'est pas du tout simulé dans la plupart des jeux, à l'exception de Hydro-

1. http ://www.realflow.com

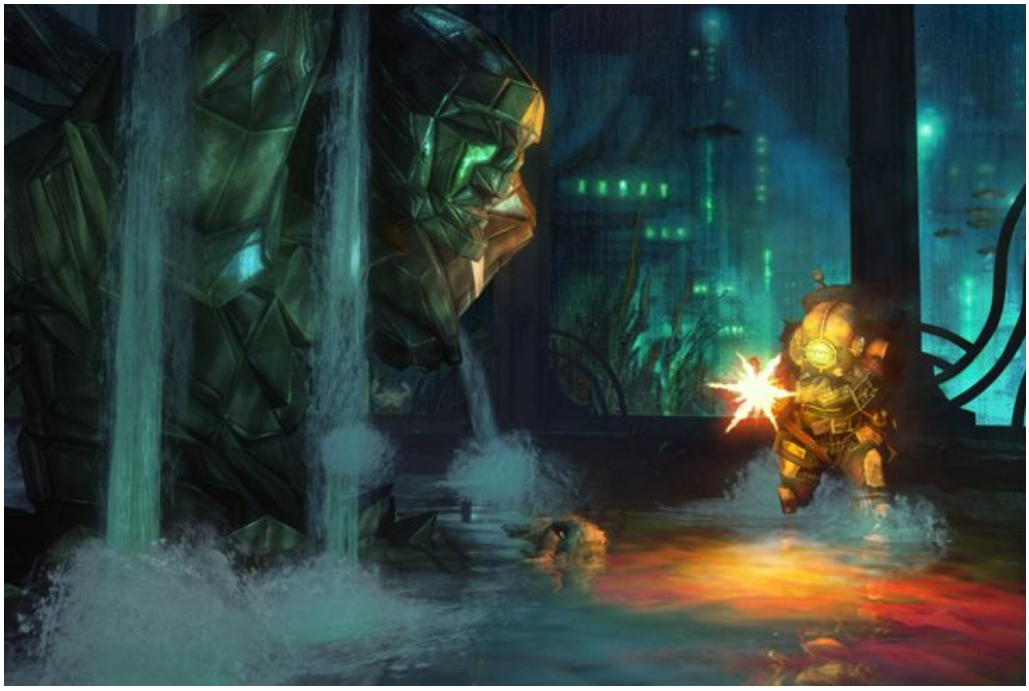


Figure 1 – Capture du jeu Bioshock se déroulant dans un environnement sous marin. Les chutes d'eau sont visiblement représentées par des aplats de textures et n'influencent pas l'eau présente dans la scène.

phobia [2] qui devrait l'utiliser comme élément de jeu. De plus, le GPU ou « Graphics Processing Unit » n'est généralement pas utilisable pour la simulation physique puisqu'il est généralement totalement dédié au traitement graphique du jeu, même si les méthodes lagrangiennes sont facilement transposables sur GPU, autorisant ainsi des simulations en temps réel.

Plusieurs éléments sont responsables du réalisme d'une simulation de fluide. Nous nous intéresserons en particulier dans ce document aux interactions fluide-solide, aux méthodes de simulation adaptatives et aux techniques de visualisation du fluide.

Interactions fluide-solide

La prise en compte des interactions fluide-solide qui détermine la réaction de l'écoulement du fluide par rapport aux objets qui l'entourent est un des facteurs indispensable du réalisme de la simulation. Il est nécessaire dans ce cadre de passer par une étape de détection de collisions qui est une étape coûteuse de la simulation. En effet plus la géométrie de la scène contient de détails plus le nombre de calculs à effectuer est élevé. Il est donc nécessaire de trouver des solutions pour simplifier ou faciliter ces calculs. Cette étape est généralement constituée de deux phases. Dans la première les objets qui sont potentiellement en collision avec les particules sont

2. Sortie courant 2010.

sélectionnés. Dans la deuxième, le calcul exact de la collision entre les particules et les objets est effectué, permettant ensuite de calculer le mouvement du fluide et des objets résultant de ces collisions.

Simulation adaptative

Les méthodes adaptatives ont été proposées afin de faire face à la demande croissante d'animations beaucoup plus détaillées. Elles permettent en effet de diminuer le coût de la simulation en allouant les ressources de calculs dans les zones où l'écoulement du fluide présente un comportement intéressant. C'est par exemple le cas des zones turbulentes, ou encore celles situées près de la surface et/ou de l'observateur qui sont les zones du fluide responsables de la qualité visuelle de la simulation.

Les méthodes adaptatives permettent d'étendre les méthodes lagrangiennes en modifiant la taille ou la distribution des particules en fonction des zones d'intérêt. Il est ainsi possible de privilégier les zones nécessitant plus de détails en utilisant une distribution plus fine de particules et d'approximer les zones nécessitant moins de détails avec des particules plus grossières sans trop sacrifier la précision de la simulation.

Visualisation du fluide

La visualisation efficace d'une simulation de fluide représente aussi un défi car il s'agit d'une des étapes les plus coûteuses et dont le résultat est directement sanctionné par les spectateurs ou joueurs. Le problème est que la surface d'un liquide n'est pas connue explicitement. Il est généralement nécessaire de calculer une surface implicite à partir des éléments obtenus lors de la simulation puis de la convertir en surface explicite afin d'en effectuer plus facilement le rendu.

L'utilisation d'une surface implicite pour représenter la surface d'un liquide est avantageuse puisque cette formulation permet de traiter naturellement les changements topologiques dus à l'écoulement du fluide. De plus, il est possible d'ajouter des effets supplémentaires comme l'effet de flou de mouvement.

Objectifs

Le constat actuel est qu'il est pour l'instant difficile de faire des compromis, il faut privilégier soit la qualité, soit l'efficacité. Il n'est pas possible d'obtenir les deux en même temps.

Un des facteurs déterminant les performances de la simulation de fluide est l'utilisation de structures de données performantes et adaptées aux différentes phases de la simulation. En effet chacune des phases décrites précédemment est basée sur

l'utilisation d'une structure d'accélération sans laquelle ces méthodes ne seraient pas utilisables. Les structures de données représentent généralement une subdivision de l'espace permettant d'associer à chaque objet une zone de l'espace réduite dans laquelle sont stockés des objets voisins.

Notre objectif est de proposer une structure de données permettant de traiter chacun de ces problèmes.

Organisation

Nous verrons tout d'abord dans le Chapitre 1 un état de l'art des méthodes de simulation de fluide, en particuliers les méthodes lagrangiennes et les méthodes adaptatives, ainsi que les structures de données nécessaires aux méthodes lagrangiennes et les méthodes permettant la visualisation du fluide. Nous présenterons ensuite dans le Chapitre 2 l'utilisation d'une structure de données hiérarchique dans le but d'améliorer le calcul des interactions fluide-solide. Dans le Chapitre 3 nous présenterons l'utilisation de ce type de structures pour les méthodes lagrangiennes adaptatives. Nous verrons enfin dans le Chapitre 4 les méthodes permettant la visualisation du fluide simulé.

Chapitre 1

État de l'art

Les méthodes de simulation de fluide sont divisées en deux catégories : les méthodes eulériennes et les méthodes lagrangiennes. Dans les méthodes eulériennes, le domaine de simulation est représenté par une grille uniforme ou tétraédrique et les quantités physiques du fluide sont discrétisées à des points fixes de cette grille (image de gauche de la Figure 1.1). Ces méthodes permettent d'obtenir des résultats de qualité mais sont généralement coûteuses. Un autre problème des méthodes eulériennes utilisant une grille de taille fixe est la difficulté de simuler des phénomènes plus fins que la taille des cellules tout en augmentant les temps de calculs, ou se résoudre à simuler des phénomènes de plus grande échelle. Les méthodes lagrangiennes ne sont pas basées sur l'utilisation d'une grille et permettent donc de simuler plus facilement des phénomènes comme les éclaboussures et les jets d'eau. Contrairement aux méthodes eulériennes les méthodes lagrangiennes évaluent les propriétés du fluide à des points mobiles de l'espace : les particules, qui se déplacent avec le fluide (image de droite de la Figure 1.1). Il est possible de voir ces particules comme des molécules de fluide.

Récemment le GPU est devenu un moyen populaire pour transformer une simulation physique coûteuse en temps de calculs en simulation temps réel. En effet les GPU actuels embarquent un grand nombre de processeurs polyvalents assemblés en parallèle et proposant une puissance de calcul autrefois réservée au super calculateurs. Cependant afin de pouvoir bénéficier de cette puissance, les méthodes de simulation doivent être fortement parallélisables. Les méthodes de simulation de fluides lagrangiennes sont généralement fortement parallélisables car elles reposent sur des opérations locales entre les particules. De ce fait beaucoup d'articles traitent de la simulation de fluide sur GPU, principalement avec la méthode « Smoothed Particle Hydrodynamics » (SPH). Les méthodes eulériennes sont elles aussi traitées sur GPU.

Les méthodes de simulation de fluides adaptatives sont actuellement des techniques populaires qui permettent d'étendre les méthodes de simulation de fluides

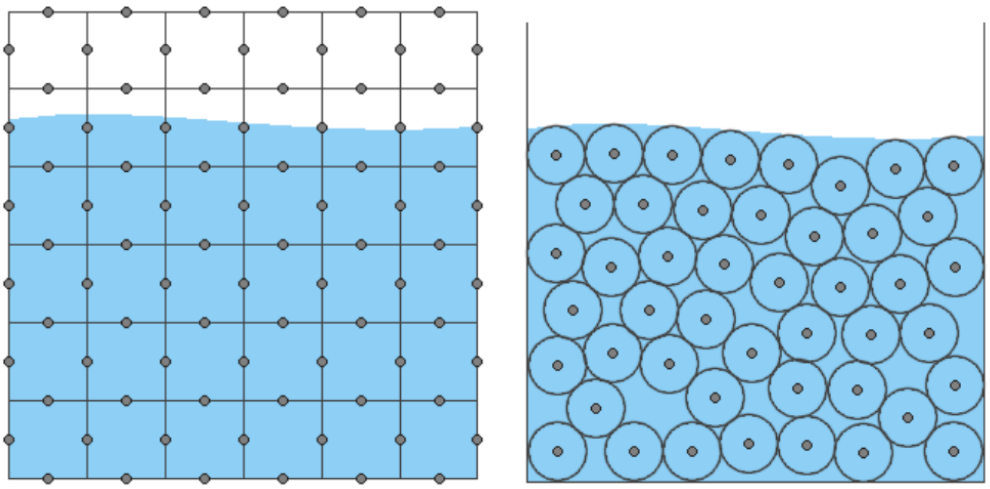


Figure 1.1 – À gauche un fluide représenté selon le point de vue eulérien. À droite le même fluide représenté selon le point de vue lagrangien (extrait de [Kel06]).

traditionnelles. Le but principal est d'augmenter la complexité des simulations produites (par exemple pour produire des jets ou gouttes d'eau) tout en minimisant les ressources utilisées. Pour les méthodes de types lagrangiennes il existe principalement deux approches. La première consiste à modifier la répartition des particules en fonction de différents critères afin de maximiser l'utilisation des ressources de calcul. L'autre approche consiste à modifier les noyaux d'interactions utilisés pour évaluer les quantités du fluide. Dans le cas des méthodes eulériennes il s'agit principalement de modifier la grille de simulation. Différents critères peuvent être utilisés afin de choisir les zones à traiter comme la profondeur dans le fluide ou encore la turbulence du fluide.

Le problème de recherche de voisinage est un problème classique des méthodes de l'informatique graphique. Il s'agit de la partie critique des méthodes de simulation de fluide lagrangiennes mais qui se retrouve dans beaucoup d'autres applications. La recherche de voisinage et la détection de collisions sont basées sur des éléments communs, notamment par l'utilisation de structures de données de partitionnement de l'espace qui sont souvent similaires. Il existe deux types de structures de partitionnement de l'espace : les structures indépendantes des objets de la scène comme les grilles uniformes, et les structures dépendantes des objets de la scène comme les structures arborescentes (KD-tree, BSP tree et octree).

La détection de collision consiste généralement à déterminer pour une trajectoire donnée si elle intersecte les tétraèdres ou polygones de la scène. C'est par exemple le cas pour l'algorithme du lancer de rayons qui consiste à trouver l'intersection des rayons primaires avec les objets de la scène. Dans le cas de la simulation d'objets déformables il s'agit de déterminer quand un objet de la scène, généralement composés

de tétraèdres, en pénètre un autre et de calculer à partir de cette pénétration la force de réponse à appliquer aux objets.

La visualisation d'un fluide liquide est une partie critique pour le réalisme d'une méthode de simulation. La visualisation est généralement composée de deux éléments : une méthode permettant de décrire la surface du fluide et une méthode permettant d'afficher la surface ainsi obtenue. Il existe dans le domaine de l'informatique graphique une grande variété de méthodes dédiées au problème de la description de surface. La raison de cette variété est qu'il n'existe pas de méthode de représentation de surfaces qui réponde aux besoins de chaque application. Dans le cas d'un fluide sa surface est généralement représentée par une surface implicite qui peut être utilisée directement pour l'étape du rendu mais qui est généralement convertie en une surface explicite plus simple à traiter.

Nous traiterons dans un premier temps les méthodes de simulation de fluide en mettant l'accent sur les méthodes lagrangiennes (Section 1.1). Nous verrons ensuite les techniques adaptatives permettant d'étendre ces méthodes (Section 1.2) et nous parlerons des structures de données nécessaires dans les méthodes lagrangiennes (Section 1.3). Enfin, nous étudierons les méthodes permettant de visualiser efficacement la surface d'un fluide (Section 1.4).

1.1 Simulation de fluides

1.1.1 Méthodes eulériennes

Les principales quantités physiques d'un fluide visqueux et isotherme sont la vélocité u, la densité massique ρ et la pression p. Ces quantités sont considérées comme étant un champ continu dans le fluide. La formulation classique de l'écoulement d'un fluide incompressible au cours du temps est décrite par les équations de Navier-Stokes :

$$\rho\left(\frac{\partial}{\partial t} + u \cdot \nabla\right) u = -\nabla p + \mu \nabla \cdot (\nabla u) + f_{ext} \tag{1.1}$$

$$\nabla \cdot u = 0 \tag{1.2}$$

où μ est la viscosité du fluide et f_{ext} représente la somme des forces de densité volumique externes comme l'attraction gravitationnelle g. Cette formulation de l'écoulement du fluide est basée sur l'évaluation des équations à des points fixes de l'espace répartis sur une structure de grille régulière, ce qui signifie que les quantités d'un champ ne dépendent pas que du temps mais aussi de la position sur la grille $r(t)$ qui elle-même dépend du temps.

L'équation 1.1 décrit la conservation du moment et est simplement la traduction de la deuxième loi de Newton pour un fluide. Cette équation indique comment le

fluide accélère à cause des forces qui s'exercent sur lui. Le membre droit de l'équation représente la somme des forces de densité volumique agissant sur le fluide et le membre gauche représente le produit de la densité massique et la densité d'accélération. L'équation 1.2 décrit la conservation de la masse d'un fluide incompressible et indique que le fluide ne doit ni gagner ni perdre de masse au cours de son évolution.

Parmi les nombreux articles simulant du fluide avec des méthodes eulériennes, l'un des plus référencés est l'article de FOSTER ET METAXAS [FM96]. Il s'agit en effet d'un des premiers articles en informatique graphique à effectuer la simulation de liquides en trois dimensions. Le liquide est simulé sur une grille uniforme pouvant contenir des obstacles entourés par le fluide. Dans un premier temps les équations de Navier-Stokes sont approximées en utilisant la méthode des différences finies sur une grille de faible résolution. Les composantes u, v et w de la vélocité sont définies au centre de chacune des faces de la cellule alors que la pression est définie au centre des cellules. La pression et la vélocité sont définies pour toutes les cellules du fluide même si la cellule contient un obstacle ou si elle est vide. La nouvelle vélocité est alors calculée pour chaque cellule en utilisant les valeurs de vélocité et de pression de l'étape précédente, il est ensuite nécessaire de vérifier que chaque cellule respecte l'équation de conservation de la masse. Pour cela, la pression à l'intérieur des cellules est ajustée selon la divergence des vélocités et la vélocité sur les faces des cellules est ajustée selon cette pression. Après cette première étape il est nécessaire de traiter le cas des cellules composant la frontière du liquide. Il s'agit principalement de l'interface entre le liquide et les obstacles et de l'interface entre le liquide et l'air. Pour simplifier la simulation, les auteurs choisissent de toujours faire coïncider les obstacles avec les faces des cellules et d'incorporer les obstacles directement dans le modèle plutôt que de les traiter comme des cas particuliers. Il est aussi nécessaire de connaître la vélocité et la pression dans les cellules vides situées à proximité de la surface libre. Pour les cellules traversées par la surface du fluide, la vélocité des faces en contact avec des cellules vides est déterminée de sorte que la divergence du fluide dans la cellule soit nulle. La pression des cellules de la surface correspond à la pression atmosphérique. Les champs de pression et de vélocité produits dans cette première étape décrivent le transport du liquide. La vélocité locale du fluide est alors utilisée pour mettre à jour un champ de hauteur ou pour transporter des particules sans masse servant à déterminer la surface libre du fluide.

STAM présente une méthode de simulation de fluide eulérienne permettant de simuler des fluides similaires à de la fumée interactivement sur des grille à deux ou trois dimensions [Sta99]. Cette méthode permet à un utilisateur d'interagir en temps réel avec le fluide. Contrairement à la méthode de FOSTER ET METAXAS [FM96] qui est eulérienne et explicite et qui nécessite des pas de temps courts pour assurer la stabilité de la simulation, la méthode de STAM est semi-lagrangienne et implicite, ce qui permet

d'utiliser des pas de temps beaucoup plus longs sans que la simulation n'explose. Il est donc possible d'augmenter le pas de temps de la simulation afin d'obtenir un résultat en temps réel. L'étape d'advection est l'étape qui est principalement responsable de la stabilité de la simulation. STAM propose donc une solution originale pour la résolution de cette étape sans contraintes sur la longueur du pas de temps. L'idée de la méthode est que à chaque pas de temps les particules du fluide se déplacent selon la vélocité du fluide lui même. Donc pour obtenir la nouvelle vélocité à un point r au nouveau pas de temps $t + \Delta t$ il suffit de regarder le parcours que ce point a effectué sur cette période Δt . La nouvelle vélocité du point r est donc la vélocité qu'il avait à son précédent emplacement au temps $t - \Delta t$. Cette ancienne vélocité est obtenue en effectuant une interpolation des vélocités voisines de cet emplacement. La deuxième différence de cette méthode par rapport à [FM96] est l'étape de projection qui doit contraindre l'incompressibilité du fluide, résolue par une étape de relaxation qui nécessite plusieurs itérations. La méthode de STAM nécessite l'utilisation d'une méthode plus précise et utilise donc la résolution de l'équation de Poisson. Une fois discrétisée spatialement l'équation de Poisson produit un système linéaire de matrices creuses qui implique la résolution de ce système linéaire. L'inconvénient principale de la méthode proposée est qu'elle souffre de beaucoup de dissipation numérique. Enfin contrairement à [FM96] toutes les quantités du fluide sont discrétisées au centre des cellules.

Beaucoup d'articles ont permis aux méthodes eulériennes de gagner en efficacité et en qualité de simulation par exemple pour limiter la dissipation numérique en utilisant une méthode d'interpolation de degré supérieur et de confinement des tourbillons [FSJ01], en ajoutant des interactions fluide-solide bidirectionnelles [GHD03] ou encore en ajoutant une viscosité variable au cours de la simulation [CMVHT02].

1.1.2 Méthodes lagrangiennes

1.1.2.1 Modéliser un fluide avec des particules

L'utilisation de particules plutôt qu'une grille fixe simplifie beaucoup les équations de Navier-Stokes. Premièrement le nombre de particules est constant et chaque particule possède une masse constante sur son volume. La conservation de la masse globale du fluide est donc garantie et la résolution de l'équation 1.2 peut être omise. Il faut cependant noter que l'incompressibilité doit être contrainte de manière locale. Ensuite l'expression $\partial u/\partial t + u \cdot \nabla u$ de l'équation 1.1 peut être remplacée par la dérivée substantielle Du/Dt. Comme les particules bougent avec le fluide, la dérivée substantielle est simplement une dérivée temporelle de la vélocité des particules et donc le terme convectif $u \cdot \nabla u$ n'est pas nécessaire dans le cas d'un système de particules.

Les trois forces (forces de densité volumique) présentes dans l'équation 1.1 modélisent les forces de pression, de viscosité et les forces externes comme l'attraction

gravitationnelle. La somme de ces forces $f = -\nabla p + \rho g + \mu \nabla^2 u$ détermine le changement du moment $\rho Du/Dt$ pour les particules. L'accélération d'une particule a_i se calcule donc avec :

$$a_i = \frac{du_i}{dt} = \frac{f_i}{\rho_i} \tag{1.3}$$

où f_i représente la somme des forces s'appliquant sur cette particule et ρ_i la densité de fluide représentée par cette particule.

Maintenant que l'ensemble des forces agissant sur les particules du fluide est connu il est nécessaire de calculer le mouvement produit par ces forces sur les particules. Chaque particule est déplacée selon un pas de temps fixe Δt en utilisant l'accélération précédemment calculée et la nouvelle position des particules est obtenue en intégrant l'accélération numériquement. Il existe plusieurs méthode d'intégration numérique comme Euler implicite et Euler explicite. Dans Euler explicite la position et la vélocité sont intégrées en parallèle :

$$r_{t+\Delta t} = r_t + \Delta t u_t \tag{1.4}$$

$$u_{t+\Delta t} = u_t + \Delta t a_t \tag{1.5}$$

La méthode Euler implicite qui est en fait une méthode semi-implicite utilise la nouvelle vélocité calculée avec l'équation 1.5 pour calculer la nouvelle position, les étapes ne sont donc plus indépendantes :

$$r_{t+\Delta t} = r_t + \Delta t u_{t+\Delta t} \tag{1.6}$$

1.1.2.2 Principes de la méthode SPH

La méthode SPH est à l'origine une méthode de simulation de phénomènes astrophysiques [Mon92], elle est cependant beaucoup utilisée dans le cadre de la simulation de fluide. Il s'agit d'une méthode d'interpolation basée sur des systèmes de particules. Avec cette méthode chaque quantité d'un champ qui est seulement défini à des points discrétisés de l'espace peut être évaluée pour n'importe quel point de l'espace. Pour cela la méthode évalue ces quantités à partir de chacune des particules du voisinage en utilisant des fonctions appelées noyaux d'interpolation. Une quantité scalaire A est interpolée à une position de l'espace r en effectuant une moyenne pondérée de la contribution de chacune des particules du voisinage :

$$A_S(r) = \sum_j m_j \frac{A_j}{\rho_j} W(r - r_j, h) \tag{1.7}$$

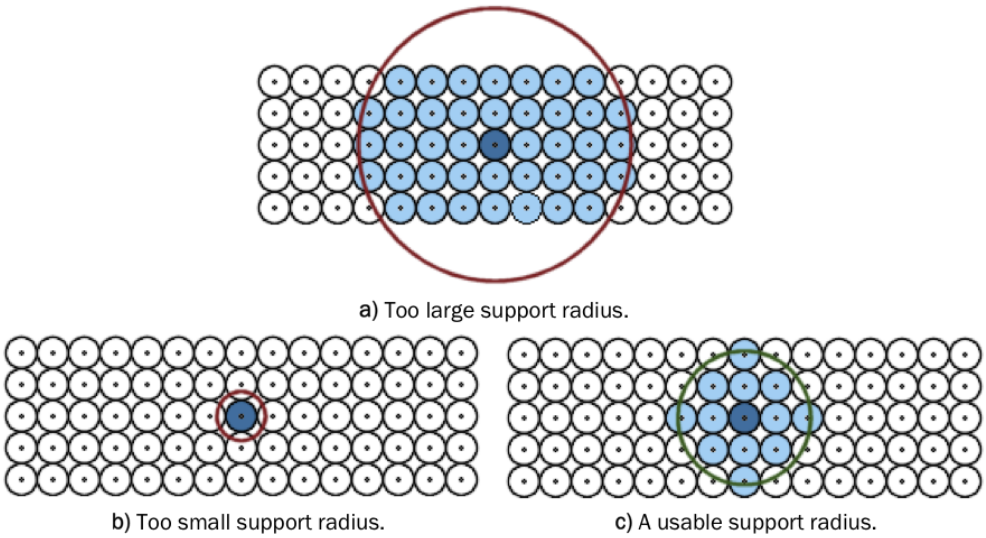


Figure 1.2 – Le rayon d'interaction d'une particule détermine la précision mais aussi la performance des calculs (extrait de [Kel06]).

où j est une particule du voisinage, m_j est la masse de cette particule, r_j sa position, ρ_j la densité et A_j la quantité scalaire à la position r_j.

La fonction $W(r, h)$ est la fonction de lissage dont le rayon d'interaction h délimite la zone dans laquelle les particules auront une influence (Figure 1.2).

La densité et la masse de la particule apparaissent dans l'équation 1.7 car chaque particule j représente un volume du fluide aproximé par $V_j = m_j/\rho_j$. Alors que la masse est constante tout au long de la simulation, la densité change et doit être évaluée à chaque pas de temps en utilisant l'équation 1.7. La densité à la position d'une particule r_i est ainsi évaluée par :

$$\rho(r_i) = \sum_j m_j \frac{\rho_j}{\rho_j} W(r_i - r_j, h) = \sum_j m_j W(r_i - r_j, h) \tag{1.8}$$

Dans les équations de simulation du fluide il est souvent nécessaire d'évaluer des gradients et des laplaciens. Dans la méthode SPH ces dérivées n'affectent que le noyau d'interpolation. Le gradient d'une quantité A est simplement évalué par :

$$\nabla A_S(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(r - r_j, h) \tag{1.9}$$

Le laplacien est évalué par :

$$\nabla^2 A_S(r) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(r - r_j, h) \tag{1.10}$$

Il est maintenant possible d'évaluer le terme responsable de la force de pression $-\nabla p$ pour chaque particule en utilisant l'équation 1.9 :

$$f_i^{pressure} = -\nabla p(r_i) = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W(r_i - r_j, h) \tag{1.11}$$

Cependant la force de pression qui est produite avec cette équation n'est pas symétrique comme on peut le voir quand seulement deux particules interagissent. Le gradient du noyau étant nul à son centre, la particule i n'utilise que la pression produite par la particule j pour calculer sa force de pression et vice versa. La pression évaluée à la position des deux particules n'est généralement pas identique, la force de pression ne sera donc pas symétrique. MÜLLER ET AL. [MCG03] proposent une méthode simple pour symétriser cette équation :

$$f_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(r_i - r_j, h) \tag{1.12}$$

Cette version est symétrique car elle utilise la moyenne arithmétique de la pression des deux particules. Les particules ne transportant que les trois quantités masse, position et vélocité il est nécessaire d'évaluer d'abord la pression à la position des particules. Cette opération est réalisée en deux étapes. Tout d'abord l'équation 1.8 nous permet d'évaluer la densité à la position des particules. Ensuite la pression est généralement calculée avec l'équation des gaz parfaits :

$$p = k\rho \tag{1.13}$$

où k est une constante du gaz qui dépend de la température. L'utilisation de cette équation ne produit que des forces répulsives, ce qui est normal dans le cas de la simulation d'un gaz qui tend à remplir le volume dans lequel il évolue. Cependant dans le cas d'un liquide comme l'eau, le fluide doit garder une cohésion interne et cherche à atteindre une densité proche de sa densité au repos. L'équation des gaz parfaits peut alors être modifiée pour créer des forces de pression qui feront tendre le fluide vers sa densité au repos ρ_0 :

$$p = k(\rho - \rho_0) \tag{1.14}$$

Le terme responsable de la force de viscosité $\mu\nabla^2 v$ peut être évalué en utilisant l'équation 1.10 :

$$f_i^{viscosity} = \mu\nabla^2 v(r_i) = \sum_j m_j \frac{vj}{\rho_j} \nabla^2 W(r_i - r_j, h) \tag{1.15}$$

Une fois encore la force qui est produite par cette équation n'est pas symétrique

parce que le champ de vélocité varie d'une particule à une autre. Comme la force de viscosité ne dépend que des différences de vélocité entre les particules et pas de la vélocité absolue, il existe une façon simple de symétriser cette équation :

$$f_i^{viscosity} = \sum_j m_j \frac{vj - v_i}{\rho_j} \nabla^2 W(r_i - r_j, h) \tag{1.16}$$

De cette manière la particule i est accélérée dans la direction de la vélocité relative de ses voisines.

Un des points importants de la méthode SPH est la fonction d'interpolation W. Cette fonction détermine la contribution d'une particule par rapport à une autre en fonction de la distance qui les sépare. Elle doit répondre à plusieurs contraintes : elle doit être normalisée, positive, symétrique et de valeur nulle au delà du rayon de support h. Le choix de cette fonction est déterminé par le problème à évaluer. Un exemple de noyau de forme gaussienne :

$$W_{gaussian}(r, h) = \frac{1}{(2\pi h^2)^{\frac{3}{2}}} e^{-(|r|^2/2h^2)},\ h > 0 \tag{1.17}$$

où $|r|$ représente la norme de r. Cependant cette fonction est coûteuse à évaluer et donc d'autres fonctions lui sont préférées.

Comme pour les autres méthodes de simulation de fluide, il est possible de rajouter un terme modélisant la tension de surface qui n'est pas prise en compte par les équations de Navier-Stokes.

La méthode SPH est notamment utilisée par Müller et al. [MCG03] qui en présentent une extension interactive dans le but de simuler des objets très déformables comme de l'eau. Afin d'atteindre cet objectif d'interactivité les auteurs proposent principalement de nouveaux noyaux d'interpolations. En particulier ils utilisent une version symétrique du calcul des forces de pression et de viscosité beaucoup plus simple qui répond à leurs besoins de rapidité et de stabilité. Les éléments qui ont le plus d'importance sur la rapidité de simulation de la méthode SPH sont les noyaux d'interpolation. Les noyaux traditionnellement utilisés sont les noyaux « B-Spline » et « Q-Spline », ils sont particulièrement coûteux car ils nécessitent l'évaluation d'une racine carrée. Müller et al. proposent, pour effectuer la plupart des calculs, d'utiliser un noyau polynomial de degré 6 qui a une forme de fonction gaussienne. Le dernier élément qui leur permet d'atteindre une simulation interactive est l'utilisation d'une grille régulière pour déterminer les particules potentiellement en interaction. Toutes ces optimisations permettent de simuler un liquide, composé de 5000 particules, qui s'écoule dans un verre à 5 images par secondes. Plus récemment, plusieurs articles proposent d'utiliser la méthode SPH sur GPU afin d'accélérer la simulation de fluide. Par exemple, Harada et al. [HKK07b] proposent une simulation dont toutes les étapes sont

effectuées sur GPU et en particulier, l'étape de recherche de voisinage qui est habituellement effectuée sur CPU.

1.1.2.3 Principes de la méthode DEM

La méthode « Discrete Element Method » (DEM) est principalement utilisée dans les domaines de la simulation de matériaux granuleux et de la dynamique des molécules [Mis03]. Le mouvement des particules est déterminé par le calcul des forces agissant entre chaque paire de particules. Ces paires de particules génèrent entre elles une force élastique f_i^{spring} modélisée par un ressort linéaire et une force d'amortissement $f_i^{damping}$ qui dissipent l'énergie entre les particules. Ces forces sont calculées pour une particule i en collision avec une particule j en utilisant les équations :

$$f_i^{spring} = -k\left(d - |r_{ij}|\right)\frac{r_{ij}}{|r_{ij}|} \tag{1.18}$$

$$f_i^{damping} = \eta v_{ij} \tag{1.19}$$

où k, η, d, r_{ij}, et v_{ij} sont respectivement les coefficients d'élasticité, d'amortissement, le diamètre de la particule, la position relative et la vélocité de la particule j par rapport à la particule i. L'équation 1.18 implique que deux particules sont en collision quand $|r_{ij}| < d$. Une force de cisaillement est modélisée comme une force proportionnelle à la vélocité tangentielle relative des particules v_{ij}^t :

$$f_i^{tangential} = k^t v_{ij}^t \tag{1.20}$$

où la vélocité tangentielle relative est calculée par :

$$v_{ij}^t = v_{ij} - \left(v_{ij} \cdot \frac{r_{ij}}{|r_{ij}|}\right)\frac{r_{ij}}{|r_{ij}|} \tag{1.21}$$

La force qui s'applique sur une particule j est la somme de ces trois forces calculées pour chacune des particules i du voisinage de j :

$$F_j = \sum_i \left(f_i^{spring} + f_i^{damping} + f_i^{tangential}\right) \tag{1.22}$$

1.1.2.4 Autres méthodes lagrangiennes

Un des avantages des méthodes lagrangiennes est l'utilisation d'opération locales qui les rend plus simples à programmer et généralement plus rapides que les méthodes eulériennes. L'inconvénient de ces méthodes est qu'elle modélisent des écoulements de fluides compressibles. Afin de pallier cette limitation il est nécessaire de trouver un équilibre entre le pas de temps, le coefficient de la force de pression et l'oscillation qui

en découle [BT07]. Plusieurs articles proposent de résoudre ce problème en utilisant des méthodes globales pour contraindre l'incompressibilité du fluide.

Premoze et al. [PTB+03] proposent d'utiliser la méthode « Moving Particle Semi-Implicit » (MPS) afin de résoudre ce problème. Tout comme la méthode SPH la méthode MPS utilise des fonctions d'interpolations (ou noyaux) pour évaluer les quantités physiques du fluide. Le problème de l'incompressibilité du fluide est traité en résolvant une équation de Poisson pour la pression qui est discrétisée en un système d'équations linéaires. Le facteur qui limite le plus les performances de la simulation est la résolution de l'équation de Poisson [1] puisque il s'agit d'une opération globale dont le temps de calcul est dépendant du nombre de particules dans la simulation.

Sin et al. [SBH09] proposent d'utiliser un diagramme de Voronoi qui est une méthode de décomposition de l'espace pour traiter le problème de l'incompressibilité. Un des avantages du diagramme de Voronoi est qu'il permet de représenter des frontières complexes en découpant les cellules de Voronoi par rapport aux obstacles et à la surface libre du fluide. La méthode se décompose comme suit : le diagramme de Voronoi est construit à partir de la position des particules, la surface libre est définie comme l'union de sphères autour des points des particules et les cellules du diagramme sont découpées par rapport à cette surface. La pression est alors projetée pour générer un champ de vélocité sans divergence et les points sont transportés à travers ce champ de vélocité. Le calcul du diagramme est cependant l'opération la plus coûteuse de la méthode.

1.1.3 Méthodes hybrides

La simulation de fluide incompressible nous confronte à deux problèmes réclamant des méthodes de simulation radicalement différentes. Le facteur dominant de l'écoulement du fluide est que le fluide transporte dans son mouvement de la matière et donc les propriétés du fluide sont transportées avec cette matière (advection). Ce comportement est similaire au point de vue lagrangien de la simulation de fluide. L'autre facteur dominant de l'écoulement du fluide est qu'il est principalement déterminé par les différences de pression. Contrairement à l'advection, la pression est un champ vectoriel dépourvu de masse qui est difficilement représenté par des particules mobiles mais plus naturellement par une grille régulière. Pour le point de vue eulérien, la façon habituelle de traiter l'advection est d'utiliser une méthode de type « retour-arrière » [Sta99]. Cependant ce type de méthode provoque une importante dissipation des zones turbulentes du fluide et donc une perte de détails visuels.

Les méthodes hybrides utilisent principalement des grilles pour calculer efficacement les interactions entre les particules nécessaires au calcul de la diffusion et pour

1. L'équation de Poisson est une équation différentielle partielle. Dans le cas de la simulation de fluide, elle permet de résoudre le problème d'incompressibilité du fluide.

contraindre l'incompressibilité du fluide (contrairement aux méthodes lagrangiennes qui utilisent des grilles pour effectuer la recherche des particules voisines).

Zhu et Bridson proposent d'utiliser les méthode hybrides « Particle-In-Cell » (PIC) [Har63] (pour les fluides visqueux comme le sable) et « FLuid-Implicit-Particle » (FLIP) [BR86] afin de simuler l'écoulement du sable. Les auteurs considèrent le sable comme un fluide fortement visqueux et utilisent donc des méthodes de simulation de fluide pour simuler son écoulement en appliquant quelques modifications afin de prendre en compte les particularités du sable. La méthode PIC est principalement une méthode eulérienne qui utilise des particules pour traiter l'advection du fluide mais toutes les autres étapes sont traitées sur la grille. À chaque pas de temps les valeurs des points de la grille sont initialisées en effectuant une moyenne pondérée de ces valeurs pour les particules situées à proximité. La grille est ensuite mise à jour jusqu'à l'étape d'advection. Les nouvelles valeurs des particules sont alors interpolées depuis la grille mise à jour, puis les particules sont déplacées selon le champ de vélocité calculé sur la grille. Le principal problème de cette méthode est qu'elle souffre d'une diffusion numérique importante due aux interpolations et moyennes répétées des quantités du fluide. La méthode FLIP permet de résoudre ce problème et permet de plus de représenter des variations importantes du fluide. La principale différence avec la méthode PIC est que la représentation du fluide est basée sur l'utilisation de particules. Les particules représentent le fluide, et la grille est utilisée uniquement pour incrémenter les valeurs des particules en fonction des changements calculés sur la grille. Cette grille n'est qu'une grille auxiliaire, en effet il n'est même pas nécessaire d'utiliser la même grille à chaque pas de temps. Il n'y a par conséquent pas de limite quant à la taille du domaine de simulation. Cette grille est entre autres utilisée pour traiter les conditions aux limites et assurer l'incompressibilité du fluide. Les auteurs précisent qu'ils est possible d'effectuer un mélange des deux méthodes afin d'ajuster la viscosité de la simulation.

Pour plus de détails sur les méthodes de simulation de fluide nous recommandons la lecture de [BMF07].

1.2 Techniques adaptatives pour la simulation de fluides

1.2.1 Adaptation de la distribution des particules

La première approche consiste à adapter la répartition et la taille des particules en fonction de différents critères. En effet, l'utilisation d'une taille de particule unique pour toute la simulation empêche la reproduction de phénomènes à plus petite échelle. Par exemple, en utilisant des grosses particules à l'intérieur du fluide et des petites particules près de sa surface il est possible d'avoir une représentation de la surface plus détaillée. Un autre critère est la distance par rapport à l'observateur, par exemple

Figure 1.3 – La méthode adaptative la plus couramment utilisée est de fusionner plusieurs petites particules en une plus grosse et inversement de subdiviser une grosse particule en plusieurs petites (extrait de [DC99]).

en utilisant des particules plus grosses loin de l'observateur puisqu'elles sont moins visibles et plus petites près de l'observateur où elles sont plus visibles. Il est enfin possible d'utiliser comme critère la turbulence du fluide. En effet l'utilisation des méthodes adaptatives peut induire des erreurs dans la simulation et lisser les aspérités du fluide. L'utilisation de particules plus petites dans les zones turbulentes permet de limiter cet effet.

La méthode de Desbrun et Cani [DC99] est une variante adaptative de la méthode SPH qu'ils utilisent pour simuler des substances déformables. Ils proposent de subdiviser les particules dans les régions où de fortes déformations ont lieu. Le pas de temps est déterminé en conséquence afin d'assurer la stabilité de la simulation. Cette méthode permet d'optimiser l'utilisation des ressources processeur tout en préservant une simulation précise de l'écoulement. La méthode adaptative retenue est donc la fusion / subdivision de particules (Figure 1.3). La fusion intervient dans des zones où la densité est stable au cours du temps. Un ensemble de particules voisines qui occupent un volume proche de celui d'une sphère est alors fusionné en une seule particule. À l'inverse la subdivision de particules intervient dans les zones où la densité varie beaucoup. Dans cette situation les particules sont subdivisées en particules plus petites. Les auteurs précisent que la productions de 7 petites particules est un bon compromis entre précision et temps de calculs. Cette méthode ne se préoccupe pas de conserver des tailles de particules homogènes : l'opération de fusion est réalisée avec des particules de tailles potentiellement différentes. Cette approche rend la méthode plus coûteuse en temps de calculs. Le deuxième apport de cette méthode est d'adapter la discrétisation temporelle en fonction de la discrétisation spatiale (les zones où la densité évolue lentement seront mise jour moins fréquemment). Ainsi l'intégration temporelle est réalisée selon un pas de temps spécifique pour chaque particule.

Adams et al. [APKG07] utilisent aussi la méthode SPH avec des techniques adaptatives. Le but est de maximiser l'utilisation des ressources de calculs en réduisant drastiquement le nombre de particules dans la simulation. Ils utilisent des critères d'échantillonnage purement géométriques et locaux qui n'affectent pas de manière significative le comportement du fluide. Ces critères permettent de réduire le nombre de particules à l'intérieur du fluide et près des zones planes de la surface (Figure

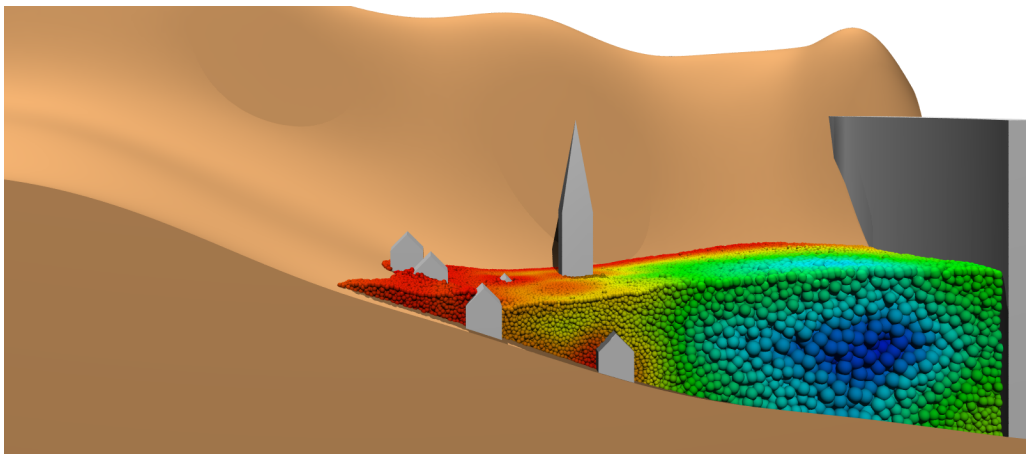


Figure 1.4 – Simulation de fluide adaptative (extrait de [APKG07]).

1.4). Ils proposent de plus d'utiliser comme critère la distance des particules par rapport à l'observateur. La recherche de voisinage des particules est effectuée grâce à un KD-tree. Cette structure n'est cependant pas mise à jour à chaque pas de temps de la simulation et les particules dont la position change dans l'arbre sont traitées séparément jusqu'à la mise à jour de l'arbre. Pour déterminer le critère d'échantillonnage adaptatif, cette méthode se base sur l'évaluation de la distance des particules par rapport à l'axe médian et à la surface du fluide (Figure 1.5). Il s'agit en fait de connaître pour chaque particule du fluide le point de la surface le plus proche et pour ce point sa distance par rapport à l'axe médian du fluide. La première étape consiste à construire l'axe médian du fluide, c'est-à-dire à déterminer quelles sont les particules les plus proches de l'axe médian. Ensuite le critère *elfs* est calculé pour les particules voisines de la surface, puis propagé récursivement aux voisines de ces particules situées à l'intérieur du fluide. Ce critère *elfs* est ensuite utilisé pour déterminer si une particule peut se subdiviser ($elfs(rx) < \alpha h_i$) ou fusionner ($elfs(r_i) > \beta h_i$). Le choix de ces valeurs permet d'éviter un processus de fusion / subdivision à répétition des mêmes particules. Lors de l'opération de subdivision une particule produit deux particules de niveau inférieur qui sont positionnées symétriquement par rapport à la particule d'origine et à une distance d de celle ci. Les nouvelles particules sont positionnées de sorte qu'elles soient à distance inférieure à d des autres particules afin d'éviter l'apparition d'une force de pression trop importante. Lors de l'opération de fusion deux particules produisent une nouvelle particule située à la position moyenne des deux particules si cette nouvelle position est à l'intérieur du volume du fluide et si la nouvelle particule est suffisamment loin des autres particules.

Zhang et al. [ZSP08] proposent une méthode de simulation de fluide utilisant une variante adaptative de la méthode SPH afin de réduire le nombre de particules dans la simulation. Cette méthode est entièrement implémentée sur GPU avec un schéma

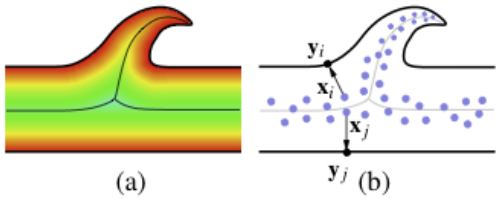


Figure 1.5 – (a) Axe médian du fluide. (b) Particules réprésentant l'axe médian du fluide. Plus une zone est proche de l'axe médian du fluide et plus elle est loin de la surface : plus cette zone peut être simplifiée (extrait de [APKG07]).

de fusion / subdivision utilisant 4 « sous »-particules. Le critère pour déterminer si une zone est géométriquement complexe consiste à calculer la distance entre une particule et le barycentre des ses voisines. Si cette distance est plus petite qu'un certain seuil δ_1, la particule est considérée comme se trouvant à l'intérieur du fluide et peut être fusionnée, si cette distance est plus grande qu'un deuxième seuil δ_2, alors elle se situe à la surface du fluide et doit donc être subdivisée si elle n'a pas déjà atteint la taille minimum. Dans le cas où la distance est comprise entre les deux seuils il n'y a aucune action à effectuer. Cette méthode permet aux auteurs de simuler un fluide composé de 50K particules à environ 20 images par seconde.

Yan et al. [YWH+09] proposent une méthode de simulation de fluide utilisant une variante adaptative de la méthode SPH réalisée sur GPU dans le but de simuler un fluide de manière réaliste et complexe en temps réel. Cette méthode propose de fusionner / subdiviser les particules du fluide selon un critère appelé fonction de champ de distance généralisé. Ce critère est en fait la somme de trois critères pondérés par des paramètres définis par l'utilisateur. Le premier critère utilisé est la distance des particules depuis la surface. Les auteurs utilisent le calcul de la normale de la surface du fluide provenant du calcul de la tension de surface pour déterminer les particules qui composent cette surface. Ensuite la distance depuis la surface est propagée par les particules de surface aux particules situées à l'intérieur du fluide. Le deuxième critère se base sur la force de pression afin d'utiliser plus de particules dans les zones de forte pression que dans les zones de basse pression. Le troisième critère permet de prendre en compte le nombre de particules situées dans le voisinage. En effet une particule disposant de peu de voisines est probablement détachée du reste du fluide. Avec cette méthode 16K particules permettent d'obtenir un résultat équivalent à une simulation utilisant 32K particules en atteignant une fréquence moyenne de 66 images par seconde.

Hong et al. [HHK08] proposent une méthode de simulation de fluide utilisant une variante adaptative de la méthode de simulation hybride FLIP [BR86]. Dans cette méthode les particules peuvent être subdivisées dans les zones de fortes déformations (principalement la surface du fluide) afin d'obtenir une représentation détaillée du

fluide ou fusionnées dans les zones nécessitant moins de détails pour une plus grande efficacité. Les auteurs utilisent un critère qu'ils appellent critère de déformabilité, basé sur l'évaluation de la profondeur des particules depuis la surface et du nombre de Reynolds[2] évalué localement. Afin d'évaluer la distance des particules depuis la surface, une représentation implicite de la surface du fluide est calculée directement à partir des particules. Ensuite un champ de distance signé est construit grâce à la méthode « Fast Sweeping » [Zha04] ce qui permet de calculer la distance depuis la surface n'importe où dans le fluide en effectuant une interpolation tri-linéaire. En utilisant ces deux critères les auteurs définissent quatre couches du fluide qui déterminent le comportement que doivent avoir les particules en entrant dans ces couches. La première couche est la couche la plus proche de la surface, toute particule entrant dans cette couche doit se subdiviser pour atteindre la taille de particule minimale. Les particules qui entrent dans la deuxième couche peuvent soit fusionner soit se subdiviser pour s'adapter à leur distance par rapport à la surface. Enfin dans les dernières couches les particules déterminent leur comportement uniquement en fonction du calcul du nombre de Reynolds. Lors de la fusion d'un groupe de particules une nouvelle particule est créée et placée au centre de gravité du groupe de particules, le volume de cette particule étant la somme des volumes de toutes les particules du groupe. Contrairement à l'opération de fusion, la subdivision consiste à produire un nouvel ensemble de particules situées à l'intérieur du rayon de la particule mère, le volume est alors réparti équitablement entre les particules filles. La position de ces particules filles est choisie aléatoirement dans le volume de la particule mère. À l'intérieur d'une couche les particules peuvent avoir des rayons différents : chacune des quatre couches dispose donc d'une taille minimale et maximale de particule. La subdivision d'une particule est interdite si les particules issues de cette subdivision sont plus petites que la borne inférieure. De même, la fusion de particules est impossible si la particule issue de cette fusion est plus grande que la borne supérieure. Ces bornes limitent en fait le processus de fusion / subdivision plutôt que la taille des particules elles-mêmes. Par exemple une petite particule peut entrer dans une couche en étant plus petite que la borne inférieure mais être incapable de fusionner sans dépasser la borne supérieure. La méthode permet de réduire d'environ 60% le nombre total de particules dans une simulation contenant environ 70K particules.

1.2.2 Adaptation du noyau

La deuxième approche consiste à adapter le noyau d'interaction des particules entre elles. Il est par exemple possible d'utiliser un noyau plus grand afin d'obtenir des interactions plus précises entre ces particules.

2. Le nombre de Reynolds est le rapport entre forces d'inertie et forces visqueuses. Il permet d'estimer la turbulence à un point donné du fluide.

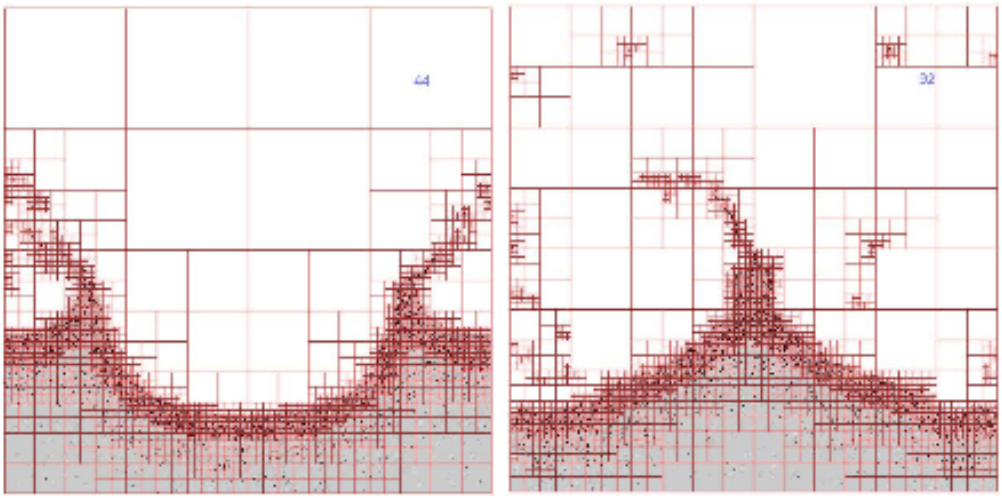


Figure 1.6 – Structure « d'octree » et particules associées (extrait de [HHK09]).

Plusieurs auteurs utilisent la méthode ASPH [OVSM98, LLL06]dans laquelle les noyaux isotropiques de forme sphérique de la méthode SPH sont remplacés par des noyaux anisotropiques de forme ellipsoïdale dont les axes évoluent automatiquement pour suivre l'espacement moyen des particules selon sa variation temporelle et spatiale, et de la direction au voisinage de chaque particule. En effet, les phénomènes gravitationnels (dynamique des gaz cosmiques, phénomènes soumis à de fortes contraintes) génèrent des changements de volumes anisotropiques, c'est-à-dire qui évoluent différemment dans chaque direction. L'utilisation de noyaux anisotropiques répond donc à cette réalité et permet, tout en gardant un nombre de particules fixe, d'augmenter la précision de la simulation pour ces phénomènes particuliers. Un des apports des auteurs est d'exprimer le changement anisotropique de volume comme une transformation locale et linéaire des coordonnées et faire ainsi paraître ce changement anisotropique comme un changement isotropique.

1.2.3 Adaptation de la grille

Les méthodes de simulation de fluide eulériennes ont connu des progrès importants privilégiant la qualité graphique plutôt que la précision numérique. Ces méthodes fournissent des résultats avec des performances impressionnantes sur des grilles 2D ou des grilles 3D plus grossières. Cependant, comme la quantité de mémoire nécessaire en 3D augmente rapidement avec la résolution de la simulation, l'utilisation d'une grille uniforme par ces méthodes empêche la simulation de phénomènes en haute résolution sur des machines conventionnelles. L'utilisation d'une grille uniforme est particulièrement pénalisante dans le cas où le volume du fluide n'occupe qu'une fraction de la grille mais peut se déplacer sur la grille entière.

Contrairement à la méthode présentée dans [HHK08], qui propose d'effectuer un changement adaptatif uniquement de la taille des particules tout en gardant une grille fixe pour les opérations eulériennes, Hong et al. proposent une version totalement adaptative de la méthode FLIP [HHK09]. Dans cette méthode un octree est utilisé afin de calculer les interactions entre particules ainsi que pour calculer les champs de vélocité et de pression (Figure 1.6). La taille des cellules de la structure d'arbre utilisée est adaptée afin de fournir une reconstruction spatiale plus fine dans les zones où les particules sont plus petites et plus grossière dans les zones où les particules sont plus grandes. Ce processus permet d'utiliser les ressources de calculs là où elles sont le plus utiles afin de conserver la complexité à la fois de l'écoulement et de la surface du fluide. L'incompressibilité du fluide est donc assurée même dans les zones très petites et turbulentes. La taille des particules est modifiée dynamiquement (en utilisant la méthode présentée dans leur précédent article [HHK08]) selon les besoins de la simulation. La structure arborescente est alors adaptée à la répartition et à la taille des particules à chaque itération de la simulation. Les auteurs précisent qu'il est judicieux d'adapter la taille des cellules de l'arbre puisque les zones qui nécessitent plus de détails dans le calcul de la pression sont généralement des zones qui nécessitent plus de détails pour le calcul de l'advection. Les particules sont d'abord placées dans un seul nœud de l'arbre. Ensuite les cellules sont subdivisées jusqu'à ce que toutes les particules qu'elles contiennent aient un rayon r inférieur à la valeur cible ($r < h/\sqrt{3}$, où h représente le côté de la cellule). De cette manière chaque particule va recouvrir une zone équivalente à huit cellules de la grille (si les cellules adjacentes étaient de la même taille). Les cellules feuilles de l'arbre qui contiennent des particules sont identifiées comme cellules du fluide. Afin d'obtenir une extrapolation précise du champ de vélocité les cellules d'air voisines des cellules de fluide de la surface sont aussi subdivisées jusqu'à ce qu'elles atteignent la taille de leur voisines contenant du fluide. Les auteurs précisent que l'utilisation d'un octree n'apporte pas de gains de temps lors de la simulation mais il permet d'obtenir des calculs de pression beaucoup plus précis dans les zones d'intérêt.

Shi et Yu [SY02] proposent d'utiliser une structure de données présentant à la fois les avantages d'une structure arborescente et ceux des grilles uniformes. Cette structure est raffinée adaptativement pour simuler l'évolution de la fumée sur des machines conventionnelles. Cette méthode subdivise adaptativement le volume de simulation en plusieurs sous-régions de sorte que plus de nœuds soient alloués dans les zones d'intérêt. Chacune des feuilles de l'arbre contient une grille uniforme qui est l'unité de base de la simulation. Les nœuds internes de l'arbre qui possèdent moins de huit nœuds fils contiennent également une grille uniforme. Seule une petite quantité de grille est conservée dans la mémoire principale de la machine, les autres grilles sont conservées sur le disque dur et rapatriées en mémoire en cas de besoin.

Les grilles elles mêmes peuvent avoir des tailles différentes, en particulier les zones produisant des variations plus importantes de vélocité ou de densité peuvent avoir une subdivision plus fine de leur grille afin de répartir efficacement les ressources de calcul. Cependant, pour plus de simplicité, tous les nœuds d'un même niveau possèdent des grilles de même taille. Cette structure évolue durant la simulation et les nœuds peuvent être dynamiquement fusionnés ou subdivisés au cours de l'évolution du fluide, principalement dans les zones soumises à de fortes variations de densité de fumée. Pour simuler un fluide sur cette structure arborescente, les équations sont résolues pour chacune des petites grilles uniformes. Il y a principalement deux étapes qui nécessitent des interactions entre les différents nœuds de l'arbre : la partie semi-lagrangienne de l'advection et la résolution de l'équation de Poisson pour la pression. L'étape semi-lagrangienne ne nécessite pas de changement tant que les cellules évaluées sont situées dans le même nœud de l'arbre. Si ce n'est pas le cas, la cellule est recherchée dans le nœud qui a la plus grande résolution pour obtenir la meilleure précision possible. L'équation de Poisson est quant à elle résolue hiérarchiquement car elle nécessite de connaître les conditions aux limites pour chacun des nœuds.

Losasso et al. présentent une méthode eulérienne adaptative qui permet de simuler de la fumée ou de l'eau en utilisant un octree [LGF04]. Cet octree permet de simuler un fluide sur un domaine qui n'a pas de frontières. Afin d'optimiser les ressources de calcul les auteurs utilisent une approche de type « niveau de détail » qui utilise plus de cellules pour traiter les zones plus intéressantes visuellement comme celles présentant des tourbillons de fumée ou des éclaboussures. Le raffinement est effectué des grandes cellules vers les petites et la simplification est effectuée des petites cellules vers les grandes. La résolution des équations du fluide nécessite une nouvelle technique pour discrétiser l'équation de Poisson sur un octree. Cette méthode produit un système linéaire non symétrique beaucoup plus complexe à inverser de manière efficace. Un des inconvénients est que l'utilisation de cellules plus larges dans l'arbre ne permet pas de simuler précisément l'interaction du fluide avec la topographie, même dans le cas où les cellules sont raffinées près du fond, les cellules plus larges situées dans la zone de l'axe médian éliminant les détails horizontaux dus au relief.

Irving et al. [IGLF06] effectuent la simulation de fluide sur une grille adaptative, basée sur l'utilisation de la technique « RLE level set » [HNB+06]. Cette nouvelle méthode permet la simulation de larges volumes d'eau en combinant des techniques de simulation en deux et trois dimensions. Le volume d'eau sous la surface est représenté par des cellules allongées en hauteur fournissant une représentation similaire à un champ de hauteur (Figure 1.7). Les cellules de la surface sont quant à elles simulées avec une méthode eulérienne en trois dimensions. L'utilisation de cellules allongées verticalement vient de la supposition que la pression varie linéairement dans la direction verticale mais pas dans la direction horizontale. Cette méthode permet de garder

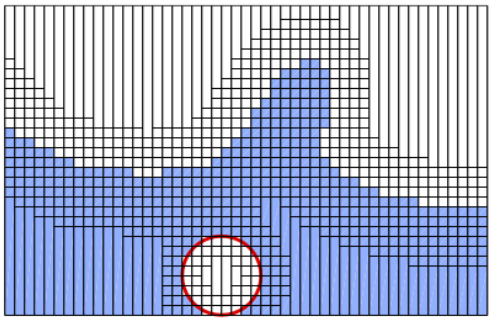


Figure 1.7 – Simulation sur une grille adaptative utilisant la méthode « RLE level set » (extrait de [IGLF06]).

une surface détaillée et de calculer précisément les déformations dues à l'écoulement sur la topographie représentant le fond. Cette méthode permet d'obtenir des gains uniquement si l'écoulement est principalement déterminé par la force de gravité, et s'effectue principalement sur l'axe horizontal plutôt que sur l'axe vertical. La structure de simulation est une grille uniforme 2D de colonnes qui peuvent contenir à la fois des cellules uniformes et des cellules allongées qui représentent un groupement de cellules uniformes. Ces cellules peuvent être réparties arbitrairement dans chaque colonne. La valeur du *level set* (méthode décrite dans la Section 1.4.1) est nécessaire uniquement dans les cellules uniformes près de l'interface du fluide alors que la pression et la vélocité sont nécessaires en tout point du fluide. Les cellules allongées contiennent des valeurs de pression correspondant aux valeurs des deux cellules situées aux extrémités du groupe de cellules qu'elle remplace. La pression est alors calculée pour les cellules intermédiaires en effectuant une interpolation linéaire entre ces deux valeurs. La vélocité est quant à elle située sur les face des cellules. Dans le cas de deux cellules allongées adjacentes la plus petite face commune aux deux contient les deux valeurs de vélocité des cellules extrêmes. Les vélocités peuvent ainsi être calculées verticalement et horizontalement par une interpolation linéaire. Les auteurs utilisent des cellules régulières à proximité des objets et dans les zones de faible « profondeur optique » par rapport à la surface, et des cellules allongées partout ailleurs.

1.3 Recherche de voisinage et détection de collisions

La partie critique dans une méthode lagrangienne est le calcul des interactions entre les particules voisines. Ces interactions sont responsables du mouvement du fluide et le problème est alors de trouver ces particules voisines aussi efficacement que possible. Ce problème est connu sous le terme recherche des plus proche voisins « Nearest Neighbors Search » (NNS). La solution la plus simple est de calculer pour chaque particule sa distance par rapport à toutes les autres particules présentes dans

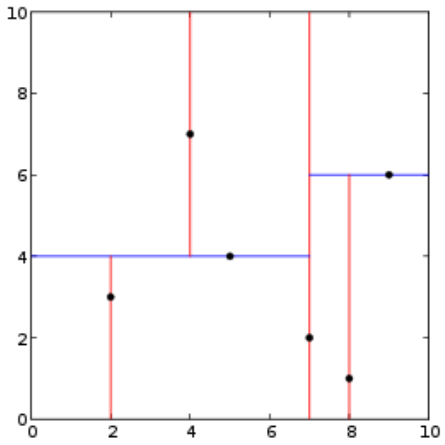


Figure 1.8 – Exemple de décomposition en KD-tree obtenue pour un ensemble de points.

la simulation. Cette solution peut suffire pour un nombre très faible de particules mais dès que ce nombre augmente cette solution n'est plus utilisable, il est donc nécessaire d'utiliser une structure d'accélération pour effectuer cette recherche de voisinage. Au cours des dernières années beaucoup de structures de données sont apparues.

L'autre partie critique dans le cas des interactions fluide-solide est la détection de collisions entre les particules du fluide et les polygones constituant la géométrie de la scène. Il s'agit de déterminer pour chaque particule un ensemble de polygones restreint qui sont potentiellement en collision avec elles. Là aussi il est nécessaire d'utiliser une structure de données adaptée à ce type de recherche.

Des structures de données identiques ou similaires sont aussi utilisées dans d'autres domaines de l'informatique graphique comme pour le lancer de rayons. Leur utilisation est cependant très différente d'une utilisation avec des particules.

1.3.1 Structures de données arborescentes

Les structures arborescentes comme le KD-tree [Ben75] sont aussi bien utilisées dans le cadre de la recherche de voisinage que dans celui de la détection de collisions. Grâce à une subdivision hiérarchique de l'espace. Chaque nœud subdivise l'espace en deux parties selon un plan aligné sur un des axes x, y et z. Chacun des trois plans est utilisé alternativement tout au long de la récursion de l'arbre. Il existe plusieurs façons de construire un KD-tree selon l'utilisation qui lui est destinée. Dans le cas de la recherche de voisinage l'arbre doit subdiviser récursivement un ensemble de points correspondant aux centre des particules (Figure 1.8). La méthode la plus simple consiste à choisir un point parmi l'ensemble de points et de l'utiliser comme pivot pour séparer

les autres points en deux sous-ensembles. Ces deux sous-ensembles sont à leur tour subdivisés récursivement. Le point qui est utilisé come pivot est généralement le point médian de l'ensemble de points par rapport au plan de coupe. La production d'un arbre équilibré est un facteur déterminant des performances de la recherche de voisinage. L'intérêt du KD tree pour la recherche de voisinage est qu'il permet d'éliminer rapidement de larges portions de l'espace et donc de tester un minimum de particules. Pour chaque nœud de l'arbre il faut tester si la sphère représentant la particule intersecte le plan de subdivision du nœud. Si c'est le cas alors la sphère d'interaction de la particule chevauche les deux parties de l'arbre, il est alors nécessaire de rechercher les particules voisines dans les deux branches de l'arbre. Dans le cas contraire il suffit de seulement vérifier la branche qui contient la particule. Le KD-tree est notamment une partie importante de l'algorithme du « photon mapping » [Jen01], il permet en effet de trouver pour chaque photon ses k voisins les plus proches et permet donc de traiter des distributions de photons non uniformes. Il faut cependant noter que le temps de construction du KD-tree comme des autres structures arborescentes empêche une utilisation dans le cas de scènes dynamiques.

Les KD tree sont en fait un cas particulier d'arbre BSP [FKN80] qui peut aussi être utilisé dans les domaines du lancer de rayons [TI08], du calcul de collision [Mel00] ou de visibilité. L'idée principale est toujours de subdiviser l'espace en deux sous-espaces, cependant les plans utilisés pour la subdivision ne sont pas nécessairement alignés sur les axes de l'espace. Les plans de subdivision sont choisis de façon à subdiviser de manière optimale les objets contenus dans l'espace. En effet un KD-tree ne sera peut-être pas capable de partitionner chaque objet dans une seule feuille puisqu'un plan de séparation aligné sur les axes n'existe peut-être pas, ce qui produit alors plus de tests d'intersection lors du parcours de l'arbre. Cependant cela rend généralement la construction et le parcours d'un arbre BSP plus complexe que pour un KD-tree qui lui est généralement préféré. Ize et al. [TI08] montrent qu'il est possible d'utiliser une arbre BSP de manière aussi efficace qu'un KD-tree dans le cas du lancer de rayons.

Le quadtree et sa version en trois dimensions l'octree sont des structures relativement proches du KD-tree. Contrairement au KD-tree cette structure n'est pas binaire, le domaine est décomposé en quatre sous domaines pour le quadtree (huit pour l'octree). Il n'y a pas de contraintes sur la taille des sous domaines même si en pratique la décomposition produit des domaines de tailles identiques. Le processus de subdivision est généralement arrêté quand le nombre d'objets dans chaque partition est en dessous d'un certain seuil ce qui produit des arbres qui ne sont pas équilibrés. La recherche d'objets dans l'arbre est similaire à une recherche dans un arbre binaire. À chaque niveau de l'arbre il faut déterminer lesquelles des branches de l'arbre correspondent à la zone de recherche. Par exemple pour déterminer les particules situées dans le rayon d'interaction d'une autre particule il faut déterminer si la zone de re-

cherche recouvre une ou plusieurs branches de l'arbre et poursuivre la recherche dans les branches correspondantes. HEGEMAN ET AL. [HCM06] utilisent par exemple un quadtree pour l'étape de recherche du voisinage d'une méthode de simulation de fluide sur GPU. La structure est reconstruite sur GPU à chaque changement de la position des particules, cependant ce processus de reconstruction ne change pas la topologie de l'arbre puisqu'elle est fixée lors de l'initialisation de cette structure. L'utilisation d'une structure ayant une topologie fixe permet de simplifier sa reconstruction et la rend donc possible sur GPU. Quand les performances de la structure se dégradent trop elle est alors reconstruite sur CPU.

1.3.2 Structures de grilles uniformes

Dans le cadre de la simulation de fluide utilisant des particules, la structure de données permettant d'effectuer la recherche de voisinage est généralement basée sur l'utilisation d'une grille uniforme [Rho92]. Le domaine de simulation du fluide est délimité par une grille de cellules à trois dimensions. Chacune des cellules de la grille est un cube dont la taille du coté est fixé avant la simulation. Chacune de ces cellules contient une liste des particules qu'elle contient. À chaque étape de la simulation chaque particule calcule la cellule dont elle est la plus proche et insère sa référence dans la liste de cette cellule. L'indice de cellule (i_x, i_y, i_z) d'une particule située au point (x, y, z) se calcule en utilisant :

$$(i_x, i_y, i_z) = (\left\lfloor \frac{x}{k} \right\rfloor, \left\lfloor \frac{y}{k} \right\rfloor, \left\lfloor \frac{z}{k} \right\rfloor) \tag{1.23}$$

où k est la taille de la cellule. La taille de la cellule est un paramètre fortement responsable des performances de la recherche de voisinage. Dans le cas où toutes les particules possèdent la même taille, l'idéal est de faire correspondre la taille de la cellule avec le diamètre des particules (ou avec le rayon d'interaction dans le cas de la méthode SPH). Après l'insertion de toutes les particules, chacune d'entre elles peut tester si elle est en interaction avec ses voisines dans la grille. Pour cela il lui suffit de tester les particules situées dans les cellules voisines de la sienne, c'est-à-dire dans un cube de 3 cellules de côté. L'avantage de la grille uniforme est que cette structure permet de garantir un temps d'insertion et d'accès aux données constant. Cependant la quantité de mémoire nécessaire augmente avec la taille du domaine de simulation ce qui peut être pénalisant pour la simulation de petites quantités de fluide pouvant se déplacer dans un grand volume. Dans les cas où l'espace mémoire est limité il est possible d'utiliser une table de hachage. Une table de hachage permet de représenter une grille uniforme virtuelle de taille infinie dans un espace mémoire limité à une dimension.

L'inconvénient majeur de la grille uniforme est qu'il lui est difficile de traiter effica-

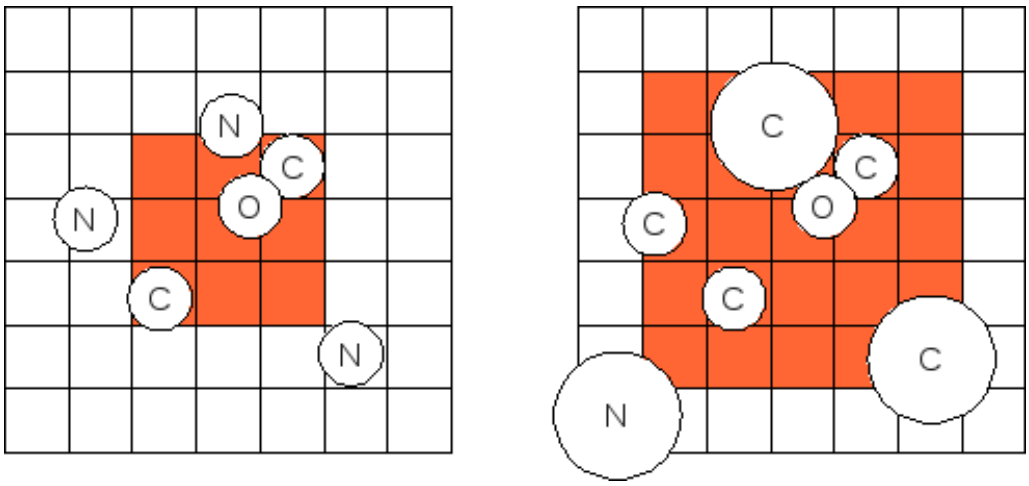


Figure 1.9 – La taille d'une particule influence son rayon de recherche dans une grille uniforme. À gauche la particule Origine recherche ses voisines dans un carré de trois cellules de côté. À droite cette même particule doit étendre son rayon afin de détecter ses voisines de plus grande taille.

cement des particules de tailles très différentes comme celles qui sont produites par les méthodes de simulation de fluide adaptatives. En effet la structure de grille uniforme repose sur l'utilisation d'une taille de cellule fixe. Si la taille d'une particule est plus grande que la taille des cellules, elle doit alors étendre son rayon de recherche afin de détecter toutes les collisions possibles. Au contraire si la taille de cette particule est plus petite que la taille des cellules alors elle va tester beaucoup plus de voisines potentielles que nécessaire (Figure 1.9).

Contrairement à la méthode traditionnelle qui consiste à insérer chaque particule dans une seule cellule de la grille et d'effectuer plusieurs requêtes sur les cellules voisines, il est possible d'effectuer ces opérations en sens inverse, c'est-à-dire d'insérer la particule dans toutes les cellules qui peuvent contenir des particules en collision. On peut ensuite effectuer une seule requête dans la table, chaque cellule contenant toutes les particules à proximité. Cette méthode peut être efficace pour une utilisation sur GPU par exemple [vKvdBT07] .

Harada et al. proposent une structure de données dédiée à la recherche de voisinage sur GPU [HKK07c] pour la simulation de phénomènes physiques représentés par des particules. Dans cette méthode une grille est construite dynamiquement à chaque étape de la simulation pour s'adapter à la distribution des particules. Le but est de fournir une structure de données permettant de simuler des phénomènes en utilisant uniquement le processeur graphique. L'utilisation d'une grille uniforme peut nécessiter beaucoup de mémoire, en particulier dans le cas 3D, les auteurs proposent donc une structure de données composée de tranches verticales du domaine de simulation qui élimine le plus possible les voxels vides et qui est donc un compromis entre facilité

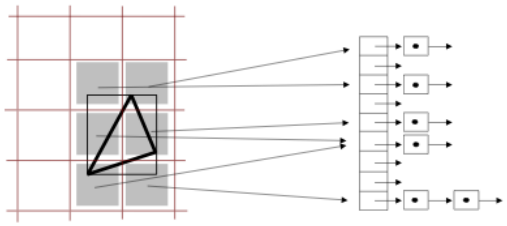


Figure 1.10 – Insertion d'un triangle dans les cellules de la table recouvertes par sa boîte englobante (extrait de [THM+03]).

de construction et besoins mémoire. Dans un premier temps chaque particule calcule l'indice du voxel auquel elle appartient dans une grille virtuelle de taille infinie. Cette grille est virtuelle car elle n'existe pas en mémoire. Chacune des tranches possède une épaisseur d'un voxel dans la direction de l'axe y et possède donc une dimension de moins que le domaine de simulation. Pour chacune des tranches 2D une boîte englobante des particules de cette tranche est calculée. Le nombre de voxels utilisé dans chaque tranche est ensuite calculé à partir de ces boîtes englobantes, le premier voxel de chaque tranche dans la mémoire globale étant obtenu en effectuant la somme du nombre de cellules pour chacune des tranches précédentes. L'utilisation d'une zone de mémoire globale contenant les voxels de chacune des boîtes englobantes permet ainsi de réduire fortement la quantité de mémoire utilisée par la structure de données : alors qu'une grille uniforme de $256 \times 256 \times 256$ voxels nécessite environ 256Mo de mémoire vive, la structure proposée ne nécessite que 15Mo pour une simulation contenant environ un million de particules. De plus, comme les voxels contenant des particules sont situés dans des zones contiguës de la mémoire les données seront plus souvent présentes dans la mémoire cache de la carte graphique. Cependant, dans le cas d'une répartition uniforme des particules dans la scène, les boîtes englobantes contiendront potentiellement beaucoup de voxels vides.

Teschner et al. proposent une structure de données adaptée à la détection de collisions pour les objets déformables [THM+03]. Les auteurs utilisent une table de hachage pour accélérer la détection de collisions entre des objets qui se déforment dynamiquement et sont composés de tétraèdres. Cette table de hachage permet de compresser une grille uniforme potentiellement infinie dans un espace mémoire fini et restreint. La reconstruction de cette table étant peu coûteuse il est possible de la reconstruire à chaque nouvelle déformation des objets ce qui est impossible avec une structure arborescente tel qu'un octree ou un arbre BSP. La première étape consiste pour chaque tétraèdre à calculer une boîte englobante, alignée sur les axes x, y et z (AABB). Chacune de ces boîtes englobantes est alors discrétisée selon la taille de cellule utilisée dans la table et pour chacune des cellules ainsi obtenues, une nouvelle

entrée est ajoutée dans la table de hachage ayant pour valeur d'indice du tétraèdre (Figure 1.10). Il s'agit de faire correspondre les coordonnées (i_x, i_y, i_z) de chaque cellule (obtenus avec l'équation 1.23) à un indice h dans la table de hachage à une dimension en utilisant la fonction de hachage $h = hash(i_x, i_y, i_z)$. La fonction de hachage utilisée par les auteurs est :

$$hash(i_x, i_y, i_z) = ((\alpha \cdot i_x) \oplus (\beta \cdot i_y) \oplus (\gamma \cdot i_z)) mod\, n \tag{1.24}$$

où α, β, γ sont de grands nombres premiers et n est la taille de la table de hachage. Lors de l'insertion d'un tétraèdre, si la table contient déjà des références vers des tétraèdres aux mêmes positions que celles du tétraèdre en cours de traitement, alors le vrai test de collision est effectué entre les deux tétraèdres. Les auteurs précisent qu'il existe plusieurs paramètres qui influent sur l'efficacité de la table de hachage. Le premier de ces paramètres est la fonction de hachage elle-même. Le choix de la fonction de hachage détermine le nombre de *collisions de hachage* : deux cellules différentes de la grille uniforme virtuelle qui sont associées à une même zone mémoire. Généralement, plus cette fonction est complexe et plus le nombre de collisions de hachage sera réduit, plus son évaluation sera longue. Le deuxième paramètre est la taille de la table de hachage. Plus la table est grande et plus le nombre de collisions de hachage se réduit. Les auteurs remarquent cependant que les fonctions de hachage sont plus efficaces avec des tables ayant une taille qui est un nombre premier. Le dernier paramètre est la taille des cellules de la grille. En effet, si la taille des cellules est beaucoup plus petite que celle des tétraèdres il faut alors utiliser beaucoup de cellules pour discrétiser les boîtes englobantes. Le nombre d'entrées à vérifier dans la table augmente également. Au contraire, si la taille des cellules est beaucoup plus grande que celle des tétraèdres, le nombre d'objets par indices de hachage augmente, de même que le nombre de tests de collisions. La taille de cellule qui permet d'obtenir les meilleurs performances est la moyenne de la longueur des bords de tous les tétraèdres. Cependant il est préférable que les tétraèdres aient une taille homogène. Cette structure de données permet aux auteurs de détecter les collisions d'environ 20K tétraèdres en temps réel.

1.3.3 Grilles hiérarchiques

Eitz et Lixu proposent eux aussi une structure de données adaptée à la détection de collisions pour les objets déformables [EL07]. Cette structure est une amélioration de la structure proposée par Teschner et al. [THM+03] et présente donc beaucoup de similarité avec cette méthode. Cette structure est une table de hachage hiérarchique qui s'adapte efficacement aux changements importants qui peuvent survenir lors de la simulation d'objets déformables (Figure 1.11). De plus il n'est, pas nécessaire pour l'utilisateur de spécifier la taille des cellules de la grille. La principale différence entre

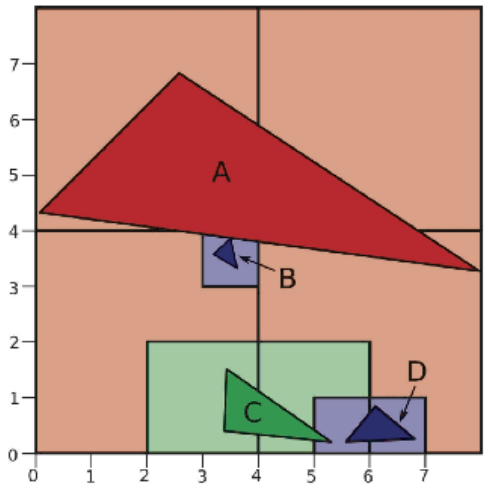


Figure 1.11 – Chacun des triangles est inséré dans les cellules du niveau le plus adapté à sa taille.

cette structure et celle de [THM+03] est qu'elle est composée de plusieurs niveaux représentant la même zone de simulation mais à des tailles différentes. Ainsi à chacun de ces niveaux est associé une taille de cellule qui est une puissance de 2 : la taille d'une cellule de niveau n est le double de celle d'une cellule de niveau $n-1$. Ces niveaux n'ont pas d'existence réelle dans la mémoire de la machine, ils agissent en fait comme un argument supplémentaire de la fonction de hachage. Il peut donc théoriquement exister une infinité de niveaux lors de la simulation. Lors de la construction de la table, chaque tétraèdre est d'abord associé au niveau de la grille hiérarchique l qui lui correspond le mieux en utilisant comme référence la taille de l'arête la plus longue de sa boîte englobante s :

$$l = \lceil log_2(s) \rceil \tag{1.25}$$

et la taille des cellules de ce niveau l est calculé par :

$$k = 2^l \tag{1.26}$$

Le but est ici d'insérer chaque tétraèdre dans le niveau qui permet d'obtenir que le nombre de cellules discrétisées de sa boîte englobante ne soit pas supérieur à 8 cellules (Figure 1.12). Après avoir calculé pour chaque tétraèdre sa boîte englobante ainsi que son niveau dans la grille hiérarchique, il est inséré dans la table de hachage en utilisant l'équation 1.23, l'équation 1.24 étant quant à elle modifiée pour prendre en compte le niveau l du tétraèdre :

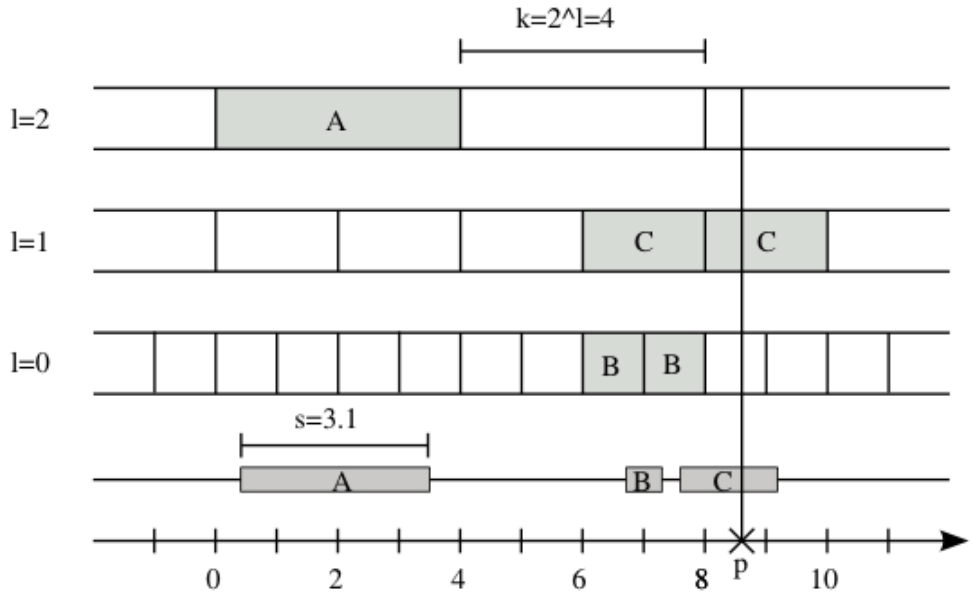


Figure 1.12 – Ce schéma représente une vue à une dimension de grilles hiérarchiques. Les objets A,B et C (en bas de l'image) sont insérés dans les cellules qui chevauchent leur boite englobante au niveau le plus adapté à leur taille. Le point p recherche pour chacun des trois niveaux de la grille les objets qui sont potentiellement en collision avec lui. (extrait de [EL07]).

$$hash(i_x, i_y, i_z, l) = ((\alpha \cdot i_x) \oplus (\beta \cdot i_y) \oplus (\gamma \cdot i_z) \oplus (\lambda \cdot l)) mod\, n \tag{1.27}$$

où $\oplus$ est l'opérateur OU exclusif (XOR).

Dans un deuxième temps les sommets des objets sont utilisés afin de déterminer les intersections entre les tétraèdres. Chaque sommet recherche dans la table de hachage les tétraèdres pour tous les niveaux présents dans la simulation. L'utilisation du niveau des tétraèdres comme argument de la fonction de hachage présente l'avantage de diminuer les collisions de hachage.

Les grilles hiérarchiques sont aussi utilisées dans le domaine du lancer de rayons. Par exemple Reinhard et al. [RSH00] utilisent des grilles hiérarchiques afin d'accélérer le rendu de scènes contenant des objets mobiles. L'utilisation d'une grille uniforme pour le lancer de rayons est intéressante même dans le cas d'une scène statique. L'inconvénient de cette structure est que plus un objet est de grande taille plus le nombre de cellules qu'il recouvre est grand et donc plus l'insertion ou la suppression de l'objet prend du temps. La solution proposée par Reinhard et al. est donc d'utiliser un ensemble de grilles hiérarchiques afin que chaque objet soit inséré dans le moins de cellules possible en l'insérant dans le niveau le plus adapté à sa taille. Le parcours de

la grille devient alors plus complexe : le rayon parcourt les cellules feuilles et chacun des nœuds parents de ces cellules jusqu'à la racine de l'arbre. Ce parcours est nécessaire car chaque niveau représente le même espace mais peut contenir des objets différents. WALD ET AL. [WIK+06] utilisent aussi des structures de grilles afin d'accélérer le lancer de rayons pour des scènes contenant des objets mobiles. Les auteurs utilisent une méthode cohérente de parcours de la grille dans laquelle les rayons sont tracés par paquets et parcourent en groupe la grille tranche par tranche plutôt que cellule par cellule. Cette méthode est de plus compatible avec l'utilisation d'une grille hiérarchique.

Pour plus de détails sur les structures de données nous recommandons la lecture de [GG98].

1.4 Construction et visualisation de la surface du fluide

1.4.1 Surfaces implicites

Une surface implicite est définie par une fonction implicite ayant une valeur scalaire sur le domaine $\mathbb{R}^3$. La surface implicite est définie par l'ensemble des points x produisant pour valeur 0 par la fonction représentant la surface. Il s'agit donc d'une fonction du type $f : \mathbb{R}^3 \to \mathbb{R}$, et la surface est représentée par :

$$\{x \in \mathbb{R}^3 | f(x) = 0\} \tag{1.28}$$

Par exemple, une sphère de rayon 1 et centrée en $(0, 0, 0)$ peut être définie par :

$$f(x) = 1 - |x| \tag{1.29}$$

Ici, l'intérieur de la surface est défini par les points ayant pour valeur $f(x) > 0$ et l'extérieur par les points ayant pour valeur $f(x) < 0$, mais cette convention peut être inversée.

Dans le cadre de la simulation de fluide, un des avantages des surfaces implicites par rapport à une représentation par maillage est qu'il n'est pas nécessaire de se soucier des problèmes de connectivité car les opérations topologiques sont gérées naturellement (par exemple la fusion de deux gouttes d'eau).

Dans le cas de la visualisation de molécules, BLINN [Bli82] propose de les représenter par un champ de densité simulant la densité d'électrons. Chaque atome est représenté par un *blob* (goutte) qui tend à se mélanger avec les *blobs* voisins s'ils sont assez proches, de la même manière que des gouttes d'eau (Figure 1.13). La valeur du champ à un point x de l'espace d'une molécule peut être calculée en effectuant la somme de la contribution individuelle des n atomes constituant la molécule à ce point :

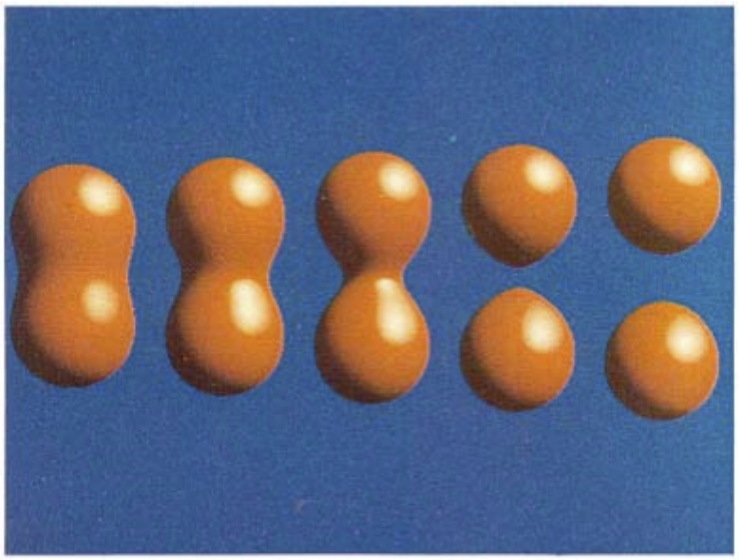


Figure 1.13 – Séparation de deux atomes (extrait de [Bli82]).

$$f(x) = -t + \sum_{i=1}^{n} h_i(x) \tag{1.30}$$

La surface est définie comme l'ensemble des points où la densité est égale à la valeur du seuil t de l'isosurface et h_i est une fonction à symétrie radiale décrivant le profil de chaque atome. Les fonctions utilisées sont généralement de forme gaussienne :

$$h_i(x) = e^{|x-c_i|^2/\sigma_i^2} \tag{1.31}$$

où la constante σ_i détermine la déviation de la fonction gaussienne et c_i représente le centre de l'atome. L'évaluation de la fonction exponentielle est une opération coûteuse possèdant de plus un support infini, plusieurs auteurs utilisent donc des expressions polynomiales pour la remplacer. Cette méthode est bien adaptée à la visualisation de densités moléculaires pour laquelle elle a été conçue. Cependant, lorsqu'elle est appliquée à la visualisation d'un grand nombre de particules macroscopiques, la surface produite est naturellement très irrégulière (même pour une surface supposée plane) ce qui met en évidence la nature de la simulation basée sur des particules plutôt que de la masquer.

Plus récemment, ZHU ET AL. [ZB05] ont proposé une méthode dont le but est de dissimuler les particules de la simulation en construisant un champ scalaire d'une manière similaire à celle de [Bli82]. Le principe de base de cette méthode est que le champ de distance signée d'une particule isolée doit être parfaitement reconstruit par la fonction implicite :

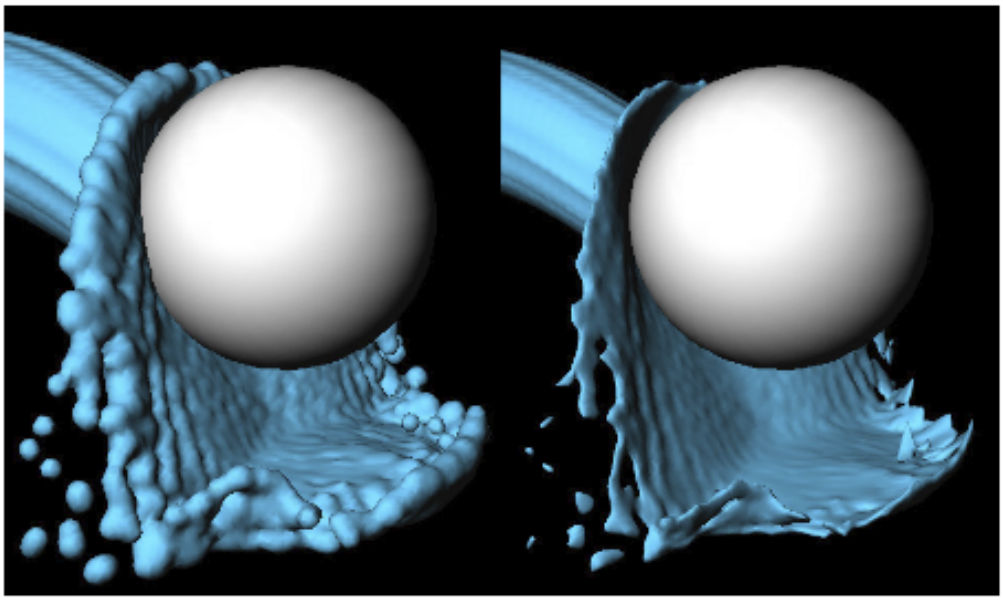


Figure 1.14 – L'image de gauche représente la surface du fluide visualisée à l'aide de *blobs*. L'image de droite représente cette surface après traitement (extrait de [HKK07a]).

$$f(x) = |x - x_0| - r_0 \tag{1.32}$$

où x_0 est le centre de la particule et r_0 son rayon. Afin de généraliser cette fonction x_0 est remplacé par une moyenne pondérée de la position des particules voisines et r_0 par une moyenne pondérée de leur rayon. Le problème de cette méthode est la création d'artefacts dans les régions concaves où des excroissances apparaissent. Ces excroissances peuvent être éliminées facilement en ajoutant une étape de suréchantillonnage puis de lissage mais au détriment des détails de surface.

Adams et al. [APKG07] reprennent la définition de surface de [ZB05] (qui en est un cas particulier) afin de suivre la distance des particules par rapport à la surface calculée durant leur simulation. Cette définition est nécessaire puisque leur méthode adaptative autorise la présence de grandes particules sur la surface du fluide permettant ainsi de limiter les artefacts dus au processus de rééchantillonnage contrairement à [ZB05]. De plus, la connaissance de la distance des particules par rapport à la surface permet de représenter beaucoup plus efficacement les surfaces planes. L'évaluation de la surface en un point se limite aux particules situées au voisinage de ce point puisque les poids de la fonction implicite ont un support compact, ce qui empêche la représentation de surfaces parfaitement planes. Enfin, cette méthode ainsi que celle de [ZB05] atténuent les détails de petite échelle.

Harada et al. [HKK07a] proposent de construire une surface présentant des

parties minces et des bords nets en utilisant des particules. Cette méthode se déroule en deux étapes : une première surface est créée par des *blobs* puis cette surface est traitée afin d'éliminer l'aspect lisse de la surface. Une surface composée de polygones est alors construite puis, pour chaque sommet du maillage ainsi créé, un nouveau sommet est calculé en effectuant une moyenne pondérée de la position des particules voisines. La position réelle des sommets est alors une valeur interpolée de sa position d'origine et de la nouvelle position. Cette opération peut être répétée jusqu'à l'obtention d'une forme de la surface suffisamment « aiguisée » (Figure 1.14). Cette méthode est cependant difficilement contrôlable puisqu'il est nécessaire d'ajuster plusieurs paramètres afin d'obtenir la surface souhaitée.

Dans le cas des méthodes eulériennes, il est nécessaire d'utiliser une méthode afin de suivre l'évolution de la surface du fluide au cours du temps. L'une des approches les plus simples consiste à utiliser des particules sans masse appelées *marker particles* dans ce but [HW65]. Ces marqueurs sont simplement utilisés pour différencier les cellules de fluide (contenant des particules) des cellules d'air (cellules vides). Le processus commence par l'échantillonnage du volume du fluide en disposant plusieurs particules de manière aléatoire par cellule. Puis les particules sont déplacées à chaque pas de temps par le champ de vélocité résultant de la simulation du fluide. Toutes les cellules de la grille contenant des particules sont alors étiquetées comme cellules du fluide. Cette information peut ensuite être utilisée dans les autres étapes de la simulation comme celle de la projection de la pression. Bien qu'elle soit très simple, cette méthode ne permet pas d'obtenir une connaissance précise de la surface du fluide. Les particules peuvent être visualisées avec une surface implicite composée de *blobs* mais toujours avec les problèmes liés à cette méthode.

La méthode des *level sets* [OF02] permet de localiser plus précisément la position du fluide. Un *level set* est en réalité une surface implicite dont la fonction f est représentée par des valeurs échantillonnées au centre des cellules de la grille de simulation. Cette caractéristique permet de faire évoluer simplement la surface avec des équations aux dérivées partielles. Contrairement à la méthode des *marker particles,* la méthode des *level sets* permet de modéliser efficacement des surfaces lisses nécessaires pour la simulation de l'écoulement de l'eau. Typiquement, la fonction f est initialisée afin qu'elle corresponde à un champ de distance signée. La distance signée d'un point x par rapport à une surface fermée est simplement la distance du point le plus proche sur la surface, avec un signe positif si x est à l'extérieur et un signe négatif si x est à l'intérieur. Cette fonction sert donc de définition pour la surface implicite du fluide. L'évaluation du champ de distance signée consiste d'abord à trouver les points de la grille qui sont proches de la surface puis pour chacun de ces points le point le plus proche de la surface. Une méthode simple [Tsa02] pour obtenir ces points est d'initialiser les cellules contenant du fluide avec la valeur -1 et les autres avec la valeur

$+1$. Les cellules qui sont proches de l'interface sont alors les cellules dont au moins une voisine a une valeur différente. La valeur de chacune des ces cellules est alors substituée par la moyenne des valeurs de ses voisines. Les points de la surface sont estimés en cherchant les points prenant la valeur zéro entre deux cellules de signes différents. Le plus proche de tous les candidats est alors retenu. La partie critique de cette méthode est le déplacement de la surface représentée par le *level set*. Les points ayant pour valeur $f = 0$ devraient être advectés par la vélocité du fluide. Cependant, l'advection d'un champ de distance signée ne préserve pas en général les propriétés de la distance signée. Il est donc nécessaire de recalculer ce champ périodiquement. Le deuxième problème lié à l'étape d'advection est la dissipation numérique qui élimine les détails les plus fin de la surface.

Afin de résoudre ces problèmes, cette méthode est généralement employée en association avec des particules disposées de chaque côté de la surface mais il reste fondamentalement impossible de représenter des détails plus petits que les cellules de la grille. Enright et al. [EMF02] proposent notamment une méthode permettant de capturer efficacement les détails d'une surface d'eau en mouvement. Cette méthode appelée « Particle Level Set », utilise des particules sans masse pour représenter la surface du fluide. Ces particules évoluent avec la vélocité du fluide et sont combinées avec une surface implicite dynamique. La méthode utilise deux ensembles de particules disposées de part et d'autre de la surface, dans une zone épaissie de la surface du fluide d'une épaisseur de trois cellules de chaque côté. Cette méthode nécessite donc d'extrapoler la vélocité du fluide à travers la surface dans la région occupée par l'air ce qui permet de mieux contrôler la surface de l'eau et de produire des effets dus au vent ou pour forcer l'eau à se stabiliser.

1.4.2 Surfaces variationnelles

La méthode des surfaces variationnelles [TO99] permet de créer une surface implicite en définissant des points de contraintes qui peuvent être de trois types : sur la surface, intérieur ou extérieur. Une fonction implicite en trois dimensions est créée à partir de ces contraintes en utilisant une méthode d'interpolation.

Les surfaces variationnelles sont apparentées aux surfaces de type *thin-plate splines* qui permettent d'interpoler un champ de hauteur à partir de points éparpillés sur un plan. D'après un ensemble de points $\{c_1, c_2, ..., c_k\}$ et de hauteurs $\{h_1, h_2, ..., h_k\}$ répartis sur un plan, il est possible de construire une surface dont la hauteur est identique à celle des points de contraintes à la même position et dont l'évolution est « lisse » entre ces points. Le résultat est une fonction scalaire $f(x)$ telle que $f(c_i) = h_i$ pour $1 \leq i \leq k$. La méthode emploie une fonction d'énergie qui mesure la qualité de la fonction d'interpolation afin qu'il existe une solution unique à ce problème :

$$E = \int_{\Omega} f_{xx}^2(x) + 2f_{xy}^2(x) + f_{yy}^2(x)dx \tag{1.33}$$

où f_{xx} représente la seconde dérivée partielle dans la direction x. Cette fonction d'énergie est en fait une mesure de la courbure globale de $f(x)$ sur une région donnée Ω. La valeur de E sera plus grande si la surface de la région considérée comporte beaucoup de replis. Au contraire si la surface est lisse et ne comporte pas de régions de forte courbure alors la valeur de E sera plus petite. La solution de la méthode *thin-plate* à ce problème d'interpolation est la fonction $f(x)$ qui satisfait toutes les contraintes et génère la plus petite valeur de E.

Une solution permettant de trouver la fonction qui permet de minimiser l'équation 1.33 est d'exprimer la solution en termes de fonctions à base radiales [3]. Par exemple, pour une interpolation en deux dimensions, l'équation 1.33 peut être résolue avec la fonction suivante :

$$\phi = |x|^2 \, log\,(|x|) \tag{1.34}$$

Dans le cas en trois dimensions la fonction généralement utilisée est :

$$\phi = |x|^3 \tag{1.35}$$

En utilisant ces fonctions à base radiales il est alors possible d'écrire une fonction d'interpolation de la forme suivante :

$$f(x) = \sum_{j=1}^{n} d_j \phi(x - c_j) + P(x) \tag{1.36}$$

où c_j représente la position des contraintes, d_j la pondération et $P(x)$ un polynôme de premier degré qui prend en compte les portions linéaires et constantes de la fonction f. La solution des poids d_j et des coefficients de $P(x)$ soumis aux contraintes produit une fonction qui interpole ces contraintes et qui minimise l'équation 1.33.

La substitution du membre droit de l'équation 1.36 par $f(c_i)$ nous donne :

$$h_i = \sum_{j=1}^{n} d_j \phi(c_i - c_j) + P(c_i) \tag{1.37}$$

Comme cette équation est linéaire, il est possible d'exprimer les fonctions d'interpolation sous la forme d'un système linéaire dans lequel $c_i = (c_i^x, c_i^y, c_i^z)$ et $\phi(c_i - c_j) = \phi_{ij}$:

3. Une fonction à base radiale (RBF) est une fonction ϕ symétrique autour d'un centre.

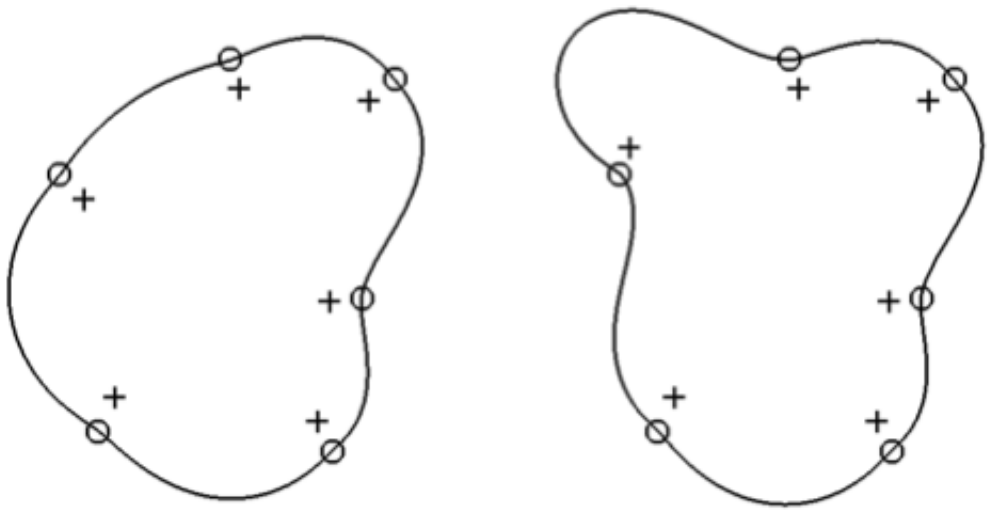


Figure 1.15 – Deux courbes implicites produites avec des contraintes positionnées sur la courbe (points zéro) identiques mais en modifiant la normale à ces points (points plus) (extrait de [TO99]).

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \cdots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \cdots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \cdots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Ce système linéaire étant semi-défini positif et symétrique, il existe une solution unique de d_i et p_i, il est alors possible d'utiliser une méthode comme celle de la décomposition LU afin de le résoudre.

La méthode de définition des contraintes est un élément déterminant la forme de la surface implicite finale. Il est possible d'utiliser plusieurs configurations de contraintes en fonction des besoins de l'utilisateur. Il est par exemple possible de placer des contraintes sur la surface et de placer une contrainte à l'intérieur de la surface, ou plusieurs contraintes à l'extérieur. Il est aussi possible de n'utiliser que des contraintes extérieures et intérieures, ou d'associer une contrainte intérieure à chaque contrainte de surface afin de représenter une approximation de la normale de la surface en ce point comme le montre la Figure 1.15.

Figure 1.16 – Transformation d'une surface en forme de croix en une surface en forme de carré (extrait de [TDOY01]).

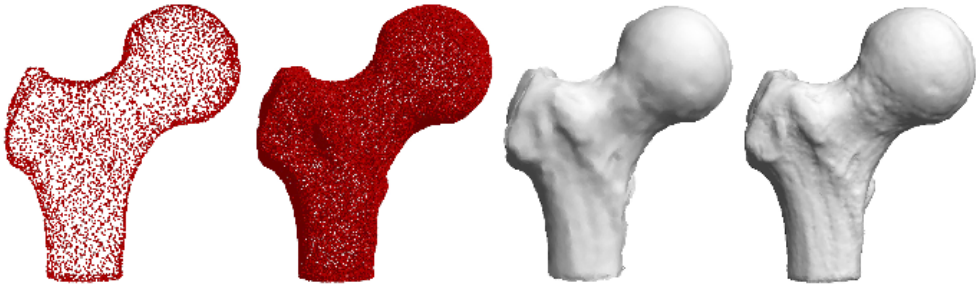


Figure 1.17 – De gauche à droite : ensemble initial des points de surface (9K), ensemble des points après suréchantillonnage (50K), surface rendue avec les points initiaux, surface rendue avec le deuxième ensemble de points (extrait de [SZP07]).

Un des avantages de la méthode des surfaces variationnelles est qu'elle n'est pas limitée à deux ou trois dimensions. Par exemple l'utilisation d'une quatrième dimension permet de représenter une évolution temporelle de la surface. De cette manière il est possible de transformer une surface implicite en une autre, c'est-à-dire d'obtenir un *morphing* (Figure 1.16) [TDOY01].

1.4.3 Visualisation

La première approche pour visualiser le résultat d'une simulation lagrangienne consiste à appliquer directement des surfels (ou aplats de texture) sur chacune des particules de surface [MCG03, ZPvBG01, vKvdBT07, SZP07]. Cette approche simple permet d'obtenir des résultats convaincants mais sous certaines conditions. Tout d'abord, le coût de la visualisation est proportionnel à la résolution de l'image finale. Bien que cette méthode soit très simple à implémenter elle nécessite un nombre important de particules afin d'obtenir une surface uniforme et régulière et afin d'éviter les effets de lissage visibles sur les bords. En effet, ce type de méthode est typiquement utilisée pour la visualisation de nuages de points obtenus à partir de scanners et contenant de 10K à 100K points à la surface de l'objet, ce qui est généralement bien supérieur au nombre de particules à la surface d'un fluide. Une solution pour les simulations de fluide est d'effectuer un suréchantillonnage des particules de surface, augmentant

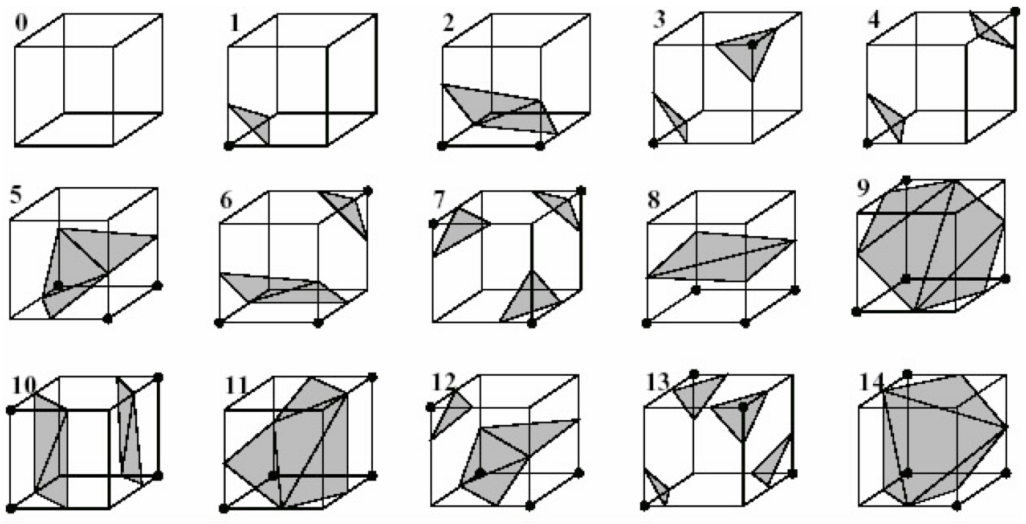


Figure 1.18 – Ce schéma présente les 15 configurations de triangles possibles (extrait de [LC87]).

ainsi la qualité du rendu. C'est la technique employée par Solenthaler et al. [SZP07] afin de produire une surface détaillée et de préserver ses éléments caractéristiques comme les arêtes (Figure 1.17). Dans cette méthode, un point supplémentaire est inséré entre chaque paire de points à chaque étape de raffinement tout en évitant d'en ajouter à proximité de ceux précédemment crées afin d'obtenir un échantillonnage quasi-uniforme. Cependant, cette méthode ne tient pas compte de la cohérence temporelle ce qui peut poser des problèmes de changements soudains de topologie (par exemple lors de la fusion de gouttes isolées).

La surface implicite est définie par une fonction continue qui n'est pas visualisable directement. Il est alors nécessaire de convertir cette surface en un ensemble de valeurs discrètes. L'isosurfaçage est le processus d'extraction d'une surface de valeur constante à partir d'un champ scalaire régulier. La méthode généralement utilisée pour cette tâche est la méthode des *marching cubes* de Lorensen et Cline [LC87], qui permet d'obtenir un ensemble de triangles approximant l'isosurface. Le but est de subdiviser le domaine en cellules cubiques et de les catégoriser selon que les valeurs des 8 sommets du cube sont supérieures ou inférieures à la valeur de l'isosurface, ce qui permet de déterminer si un cube se situe à l'intérieur, à l'extérieur ou est traversé par l'isosurface. Dans ce dernier cas, les valeurs de ses sommets permettent de connaître les arêtes concernées. Une arête est traversée lorsque les valeurs aux extrémités sont de signes opposés. Il existe en tout 2^8 configurations possibles pour chaque cube, réductibles à 15 grâce aux symétries par réflexion et par rotation. Chacune de ces 15 configurations génère une configuration de triangles représentant la surface interpolée qui sont détaillées sur la Figure 1.18. Les sommets des triangles sont interpolés linéairement en fonction des valeurs aux sommets du cube afin que la position des triangles

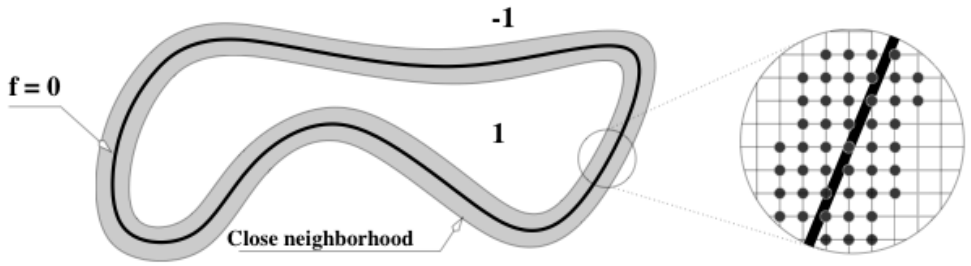


Figure 1.19 – Ce schéma 2D présente les points du champ discret situés à proximité de l'isosurface (extrait de [DC99]).

correspondent le plus possible à la position de la surface. Le maillage produit par cette méthode peut ensuite être affiché simplement avec toutes les méthode utilisant des triangles comme le lancer de rayons ou OpenGL. Il est cependant nécessaire de citer certains inconvénients de cette méthode. En effet certaines configurations sont ambigües et peuvent produire des trous dans le modèle, ou plus généralement produire une surface qui n'est pas une variété. Il existe plusieurs solutions afin de résoudre ces ambiguïtés comme utiliser des informations sur les cubes adjacents afin de vérifier que le résultat produit correspond. Il est encore possible de subdiviser les cubes jusqu'à disparition des ambiguïtés. Cependant ces solutions sont souvent coûteuses à utiliser c'est pourquoi plusieurs variantes ont été proposées en coupant les cellules cubiques en tétraèdres [MW97], en considérant les faces plutôt que les cubes [cHcWyCO05] ou encore en utilisant des octaèdres [CTM02].

Un des problèmes de la méthode des *marching cubes* dans le cas du rendu de fluide est le coût important de l'évaluation de la fonction implicite aux sommets des cubes, puisqu'il est nécessaire d'évaluer de manière exhaustive la totalité des cubes. En effet cette méthode a été conçue pour traiter les données produites par les appareils de scanner et d'IRM qui ne nécessitent donc pas l'évaluation d'une fonction implicite. Dans [DC99] Desbrun et Cani utilisent une fonction implicite discrète pour représenter la surface du fluide afin d'optimiser le processus d'extraction de la surface. Cette surface est modélisée par l'isosurface d'un champ discret dont les valeurs sont stockées dans une grille à trois dimensions. Les nœuds de la grille situés près de la surface sont stockés dans une liste mise à jour durant l'animation (Figure 1.19), les valeurs de ces points sont ensuite modifiées dynamiquement à chaque pas de temps afin d'animer la surface en fonction de la vélocité du fluide grâce à la méthode des différences finies. L'étape de polygonisation est alors simplifiée puisque les voxels intersectant la surface font partie de ceux contenus dans la liste.

Le lancer de rayons est une méthode permettant de visualiser une surface implicite directement sans la convertir sous forme de maillage. Cette approche a l'avantage

d'offrir un rendu de meilleur qualité puisque la surface composée de triangles n'est qu'une approximation de la surface réelle et peut nécessiter une discrétisation très fine et très coûteuse pour le même niveau de qualité. La partie critique est alors la détermination des intersections entre les rayons et la surface implicite qui consiste à résoudre la fonction implicite pour chaque rayon. Cette opération est simple pour certaines surfaces mais d'autres ne permettent pas d'obtenir directement le point le plus proche. Plusieurs solutions permettent de palier ce problème. L'utilisation d'une méthode de partitionnement de l'espace est fréquente, par exemple l'échantillonnage de la fonction implicite sur une grille régulière permet ensuite au rayon d'avancer pas à pas dans la grille jusqu'à atteindre la surface. Il est aussi possible d'utiliser la méthode itérative de Newton qui évalue la dérivée de la fonction implicite à chaque itération. Pour plus de détails sur les méthodes de visualisation des surfaces implicites nous recommandons [BW97].

1.5 Conclusion

Les méthodes de simulation de fluide ont été l'objet de beaucoup d'articles ces dernières années, aussi bien pour les méthodes eulériennes que pour les méthodes lagrangiennes. Cependant, la frontière entre ces deux classes de méthodes est de plus en plus floue. En effet, les méthodes lagrangiennes nécessitent l'usage de structures de données (souvent des grilles uniformes) pour calculer les interactions entre particules voisines ou pour contraindre l'incompressibilité du fluide. De même, les méthodes eulériennes utilisent des particules pour contrer les problèmes de dissipation des zones turbulentes inhérents à ces méthodes ainsi que pour effectuer la visualisation du fluide lui même. De fait les méthodes de simulation de fluide sont maintenant plus souvent des méthodes hybrides. Ces méthodes restent cependant coûteuses en ressources de calcul et sont difficilement utilisables dans les applications interactives souhaitant simuler l'écoulement de fluides en trois dimensions. Les méthodes lagrangiennes sont généralement préférées dans ces cas car elle représentent un bon compromis entre ressources nécessaires et qualité visuelle. Un des problèmes reste la prise en compte des réactions des particules par rapport à leur environnement qui nécessite d'effectuer beaucoup de calculs de collisions. L'utilisation de méthodes adaptatives, particulièrement pour les méthodes lagrangiennes, est apparue comme une solution intéressante pour augmenter le réalisme de la simulation ou encore pour diminuer son coût en temps de calcul, cependant l'opération de recherche des particules voisines devient une opération plus coûteuse avec les structures de données traditionnelles.

La création de la surface implicite du fluide est généralement une opération nécessaire à sa visualisation. Les méthodes reconstruisant la surface avec des *blobs* sont les plus utilisées et sont souvent modifiées afin d'obtenir des surfaces plus lisses ou

présentant plus de détails. Ces méthodes peuvent être améliorées en ne prenant en compte que les particules de surface. En effet, l'évaluation du champ scalaire pour l'ensemble des particules peut être très coûteux. De plus la contribution des particules les plus éloignées est négligeable et peut donc être ignorée. Cependant il peut être difficile d'identifier parfaitement les particules de surface, ce qui produit une distribution de particules irrégulière et peut laisser des trous dans la surface finale. Les surface variationnelles n'ont, à notre connaissance pas été, utilisées pour représenter des fluides, cependant leur capacité à traiter efficacement des distributions de points irrégulières semble idéale pour les méthodes lagrangiennes. L'application des techniques de morphing aux méthodes de simulation de fluides représente une utilisation intéressante de ce type de surfaces. Le plus souvent, la visualisation de la surface elle-même repose sur la construction d'une surface explicite avec une méthode de polygonisation. Il est toutefois préférable de combiner ces méthodes à des structures de partitionnement de l'espace afin d'accélérer la création des triangles dans les au zones proches de la surface.

L'utilisation de structures de données efficaces est un élément primordial des méthodes de simulation lagrangiennes d'autant plus si l'environnement du fluide est composé d'objets de taille très différentes ou si les particules du fluide peuvent elles mêmes changer de taille. Nous nous intéresserons donc dans les chapitres suivants aux solutions permettant d'améliorer les méthodes de simulation lagrangiennes en proposant des structures de données hiérarchiques adaptées aux interactions fluide-solide (Chapitre 2) et à la recherche de voisinage dans le cas d'une simulation adaptative (Chapitre 3). Nous proposerons enfin une méthode de visualisation adaptée aux méthodes lagrangiennes et décrirons l'utilisation des structures hiérarchiques dans ce cadre (Chapitre 4).

Chapitre 2

Grilles hiérarchiques pour l'interaction fluide solide

La simulation de fluides lagrangiens et la simulation d'objet solides sont généralement traitées séparément puis rassemblées lors de l'étape de détection de collisions. Les principaux problèmes à traiter pour ce type de simulation sont : le calcul des interactions entre particules voisines et la détection de collisions de particules avec les objets représentant l'environnement de la scène. Ces deux étapes nécessitent l'utilisation de structures de données permettant d'accélérer la recherche des particules voisines et l'obtention des objets potentiellement en collision avec les particules. Il n'existe généralement aucune relation entre les structures utilisées dans ces deux phases alors qu'elles représentent des éléments évoluants dans la même scène. Ces deux structures sont redondantes et augmentent la consommation mémoire. Nous proposons donc d'utiliser une structure unifiée afin de traiter ces deux problèmes en utilisant une structure de grilles hiérarchiques.

La détection de collisions est généralement constituée de deux étapes. Dans la première étape dite « approximative » le but est d'éliminer rapidement un grand nombre de polygones qui ne sont pas en collision avec une particule. Dans cette étape une structure de données permet de trouver facilement les polygones des objets qui ne sont visiblement pas en collision. Dans la deuxième phase dite « exacte » les polygones restants sont testés pour déterminer s'il existe réellement une collision avec la particule et ainsi déterminer la réponse à appliquer.

Les particules et les objets de la scène étant en mouvement, il est nécessaire de mettre à jour les informations de voisinage à chaque pas de temps de la simulation, ce qui est une partie très coûteuse en temps de calcul dans la simulation. Il est donc nécessaire de fournir une structure de données efficace pour traiter le cas d'objets fortement mobiles ou déformables. Notre méthode est une extension de la structure de hachage multi-niveaux décrite dans [EL07] dans laquelle les relations entre cellules

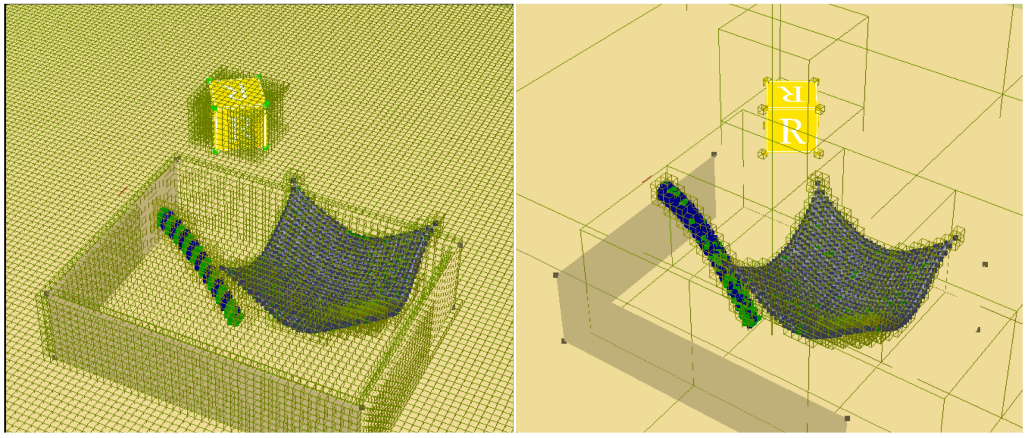


Figure 2.1 – Une scène « rastérisée » avec une taille de cellule unique (à gauche) et en utilisant un modèle multi-niveaux (à droite).

et sous-cellules sont stockées explicitement. Le choix de cette structure nous semble judicieux car elle permet de traiter des objets de tailles très différentes. Cette structure de hachage hiérarchique permet de réduire drastiquement les calculs à effectuer dans les scènes contenant de vastes zones vides et donne donc de meilleurs résultats que les méthodes traditionnelles. De plus un schéma de hachage hiérarchique permet de réduire fortement les collisions de hachages qui pénalisent habituellement les structures de hachages. Cette méthode cumule donc les avantages des tables de hachages et des structures arborescentes comme le KD-tree.

Les travaux présentés dans ce chapitre ont fait l'objet d'une publication dans la revue *journal of graphics, gpu, and game tools* [PACT09].

2.1 Hachage hiérarchique

Le principal intérêt de l'utilisation d'une structure hiérarchique dans notre cas est de pouvoir associer à chaque objet un niveau de hachage en fonction de sa taille afin de limiter le nombre d'insertions dans la structure (Figure 2.1).

2.1.1 Collisions de hachage

Les collisions de hachage représentent un problème bien connu des tables de hachage à ne pas confondre avec la détection de collisions. La structure interne d'une table de hachage est souvent représentée par un tableau : c'est-à-dire une zone de mémoire contigüe de taille généralement fixe. Cette zone de mémoire limitée représente une grille virtuelle de taille potentiellement infinie, ou plus généralement beaucoup plus grande que la zone qui doit la contenir. Il arrive alors que des objets situés dans des cellules différentes de la grille soient insérés dans la même zone mémoire de la

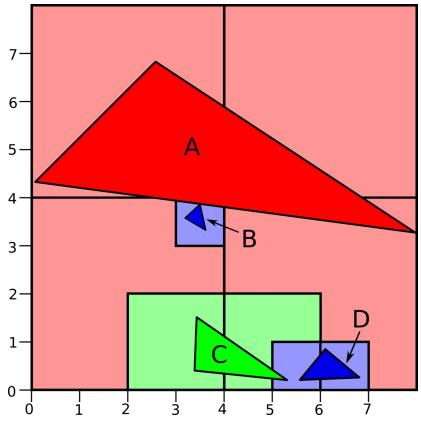


Figure 2.2 – Le schéma montre un exemple 2D dans lequel des objets de différentes tailles sont insérés dans les classes de hachage correspondantes : 0, 1 et 2 (qui correspondent aux tailles de cellule 1, 2 et 4). Les boites englobantes des objets ne sont pas représentées.

structure ce qui augmente alors le nombre de candidats pour la détection de collisions ou la recherche de voisinage. En effet la clé de hachage h attribuée à chaque objet est calculée par :

$$h^c(p) = ((\alpha \cdot i_x) \oplus (\beta \cdot i_y) \oplus (\gamma \cdot i_z) \oplus (\lambda \cdot c)) mod\, n \tag{2.1}$$

où c est la classe de hachage de l'objet, $i_x, i_y i_z$ sont les coordonnées de la cellule de la grille contenant le point p et n est la taille de la mémoire utilisée par la table de hachage. L'opération de modulo est donc responsable des collisions de hachage. Il est important de voir que ces faux candidats ne changent pas le résultat de la simulation mais qu'il augmentent cependant son temps de calcul. Le Tableau 2.1 montre par exemple des collisions de hachage obtenues avec la méthode de Eitz et Lixu avec les objets de la Figure 2.2.

2.1.2 Initialisation des classes de hachage

Contrairement à la méthode de Eitz et Lixu, notre méthode nécessite de précalculer l'ensemble des classes des objets de la scène lors de la phase d'initialisation. En effet notre méthode de hachage utilise des informations hiérarchiques pour diminuer les collisions de hachage. La taille de classe minimale dans la simulation est en général la taille r du rayon d'interaction des particules du fluide. Comme la simulation de fluide requiert généralement que cette valeur soit la plus petite possible, il n'est pas imprudent de supposer qu'aucun objet de la scène ne sera plus petit que cette valeur r.

HashKey	Eitz et Lixu	Notre Méthode
0	D	D
1	C, A	A
2		
3		
4	A	A
5		C
6	A, B	A,C
7	A	A
8		D
9		
10	D	
11		B
12		
13		
14	C	

Eitz et Lixu
$h^0(3,3) = ((\frac{3}{1} \cdot 5) \oplus (\frac{3}{1} \cdot 3) \oplus (0 \cdot 11)) = 6$
$h^0(3,3) \bmod 15 = 6$

Notre Méthode
$h^0(3,3) = ((\frac{3}{1} \cdot 5) \oplus (\frac{3}{1} \cdot 3) \oplus (0 \cdot 11)) = 6$
$h^1(3,3) = ((\frac{3}{2} \cdot 5) \oplus (\frac{3}{2} \cdot 3) \oplus (1 \cdot 11)) = 13$
$h^2(3,3) = ((\frac{3}{4} \cdot 5) \oplus (\frac{3}{4} \cdot 3) \oplus (2 \cdot 11)) = 22$
$(h^0(3,3) + h^1(3,3) + h^2(3,3)) \bmod 15 = 11$

Table 2.1 – Le premier tableau présente le hachage des objets avec la méthode de Eitz et Lixu et avec notre méthode. La taille n de la table est de 15 cellules et les nombres premiers utilisés dans la fonction de hachage sont $\alpha = 5$, $\beta = 3$ et $\gamma = 11$. Bien que cet exemple soit simple, les collisions de hachage apparaissent dans les deux méthodes même si le hachage hiérarchique permet de les diminuer légèrement. Les deux tableaux suivants présentent le détail du calcul des clés de hachage de l'objet B pour les deux méthodes.

Multilevel hashing	
Hashkey	Objects
$h^2(0,0)$	A
$h^1(2,0)$	C
$h^2(4,0)$	A
$h^2(0,4)$	A
$h^1(4,0)$	C
$h^2(4,4)$	A
$h^0(3,3)$	B
$h^0(5,0)$	D
$h^0(6,0)$	D

Hierarchical hashing		
Hashkey	Objects	hasChildren
$\tilde{h}^2(0,0)$	A	true
$\tilde{h}^1(2,0)$	C	false
$\tilde{h}^2(4,0)$	A	true
$\tilde{h}^2(0,4)$	A	false
$\tilde{h}^1(2,2)$		true
$\tilde{h}^2(4,4)$	A	false
$\tilde{h}^1(4,0)$	C	true
$\tilde{h}^1(6,0)$		true
$\tilde{h}^0(3,3)$	B	
$\tilde{h}^0(5,0)$	D	
$\tilde{h}^0(6,0)$	D	

Table 2.2 – Le tableau de gauche montre les cellules obtenues avec la fonction de hachage de EITZ ET LIXU. Le tableau de droite montre les cellules obtenues avec notre fonction de hachage hiérarchique.

2.1.3 Hachage descendant

Afin de bénéficier de la cohérence de la répartition hiérarchique des objets, la fonction de hachage définie par l'équation 2.1 est étendue de manière hiérarchique. Chaque élément (objet ou particule) utilise cette fonction de façon descendante : de la plus grande classe k présente dans la simulation, jusqu'à la classe c correspondant à la classe de cet objet obtenue avec l'équation 1.25. L'intervalle de classes parcouru est donc $[k..c]$, où $c \in [0..k]$ et la nouvelle fonction de hachage pour chaque point p est donnée par :

$$\widetilde{h}^c(p) = \Big(h^k(p) + h^{k-1}(p) + ... + h^c(p)\Big) \, mod \, n \tag{2.2}$$

L'élément peut alors être inséré dans la cellule correspondant à la clé $\tilde{h}^c$ obtenue avec cette fonction. Pour chaque classe traversée, la clé de hachage de la classe est calculée et ajoutée à celles précédemment calculées et une variable booléenne est stockée dans la table de hachage pour indiquer si la cellule possède au moins une sous cellule non vide, c'est-à-dire si une cellule existe dans une classe inférieure qui chevauche $\tilde{h}^c$ et qui contient au moins un élément. L'Algorithme 2.1 présente l'algorithme correspondant à ce processus. Le Tableau 2.2 montre l'utilisation des fonctions de hachage hiérarchiques avec les objets de la Figure 2.2.

Algorithme 2.1 Hachage descendant d'un point.

```
void insert(point p, int id, int size) {
 int hashKey = 0;
 for each (int currSize in classSizes) {
  hashKey += h(p, currSize);
  if (size == currSize) {
   hm[hashKey].insert(id); break;
  }
  else if (! hm[hashKey].hasChildren)
   hm[hashKey].hasChildren = true;
 }
}
```

Cette structure de données est donc basée sur l'utilisation de deux structures :

- une table de hachage *hm* contenant pour chaque clé de hachage, une liste des références de chacun des objets contenus dans la cellule correspondante ainsi qu'une variable booléenne *hasChildren*,
- un ensemble de *classSizes*, contenant l'ensemble des tailles de classes présentes dans la structure qui sont déterminées lors d'une étape de précalculs et triées dans un ordre décroissant.

Bien qu'il soit nécessaire d'effectuer plusieurs appels à la fonction de hachage h, la fonction de hachage additive décrite dans l'équation 2.2 nous permet d'éliminer des calculs ultérieurs coûteux et inutiles en réduisant les collisions de hachage (Tableau 2.1). Considérons deux points de l'espace $p1, p2$ qui ne sont pas dans la même cellule de l'espace de classe $c+1$: $h^{c+1}(p1) \neq h^{c+1}(p2)$. Supposons maintenant qu'il se produise une collision de hachage lors du hachage de ces points dans la classe c : $h^c(p1) = h^c(p2)$. Cette collision est évitée en utilisant l'équation 2.2 car $h^{c+1}(p1) + h^c(p1) \neq h^{c+1}(p2) + h^c(p2)$. Deux points situés dans la même cellule finale doivent obtenir la même clé de hachage et donc le même décalage produit par les appels successifs à la fonction de hachage pour les deux, ce qui est vrai dans notre cas grâce à la cohérence hiérarchique de la structure. En effet, deux points situés dans la même cellule ont aussi les même cellules parentes, les classes parcourues par la fonction de hachage sont donc les mêmes pour les deux points. Les collisions de hachage peuvent toujours se produire puisque rien ne garantit que la nouvelle cellule soit libre, toutefois cette méthode de hachage donne clairement de meilleurs résultats qu'une méthode de hachage simple. Parmi les éléments déterminant les performances de structures de hachage en général, il est aussi important de noter que l'augmentation de la taille de la table de hachage contribue à diminuer les collisions de hachage ; néanmoins plus de mémoire est alors nécessaire.

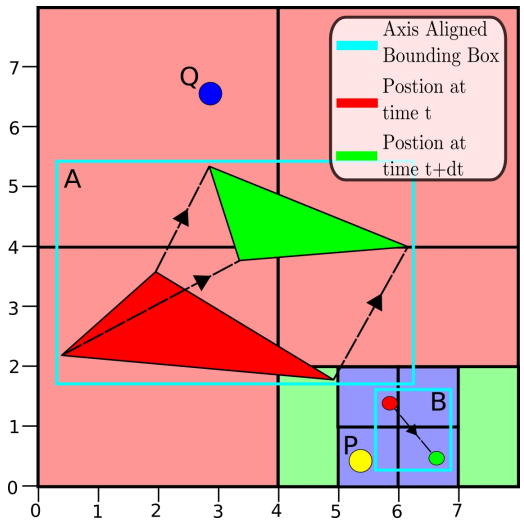
2.1.4 Insertion de particules et d'objets

L'insertion d'une particule dans la table de hachage est directe : la position de la particule est utilisée pour calculer les clés de hachage et le rayon d'interaction r pour chaque classe de hachage. Comme r est aussi la plus petite classe de *classSizes*, chacune des classes stockées dans *classSizes* est utilisée pour calculer la clé de hachage de chaque particule. La méthode d'insertion d'une particule est similaire à celle, plus générale, de l'insertion d'un point (Algorithme 2.1) où le paramètre *size* prend pour valeur r. L'insertion d'un objet plus large comme un polygone n'est pas beaucoup plus compliquée. Dans un premier temps la boite AABB (boite englobante alignée sur les axes du repère) de l'objet doit être calculée ainsi que la classe de hachage c dans laquelle l'objet doit être inséré. Ensuite le hachage descendant est appliqué pour chacun des huit points de la boite englobante.

Cette approche peut être utilisée pour insérer des objets statiques ainsi que des objets mobiles. En supposant que les objets aient un déplacement linéaire au cours d'un pas de temps d'intégration, nous considérons simplement la boite englobante de la trajectoire complète de l'objet : par exemple, la trajectoire du déplacement d'un triangle est un prisme. Il suffit alors d'insérer la boite englobante obtenue à partir de la trajectoire de l'objet de la même manière que les objets immobiles (Figure 2.3). Contrairement à d'autres structures de données cette méthode permet d'insérer les éléments en mouvement à chaque pas de temps peu importe la distance parcourue par ces éléments.

2.1.5 Requête sur la structure de hachage

Chaque requête dans la structure de hachage profite de la méthode de hachage descendante des objets statiques ou mobiles présentée dans la section précédente. Comme le montre la Figure 2.3, l'opération permettant de trouver pour un point de l'espace tous les candidats qui sont potentiellement en collision ou interagissent avec ce point, consiste à hacher ce point de manière hiérarchique pour chacune des classes depuis la plus grande et d'obtenir les références des candidats stockées dans les cellules correspondantes. Ce processus s'arrête lorsque la variable booléenne de la cellule courante indique que les cellules filles ne contiennent pas d'objets, ou si la plus petite classe est atteinte. Cependant le problème est un peu plus complexe pour la recherche de voisinage ou pour la phase large de la détection de collision. Cette méthode simple est étendue dans les sections suivantes pour traiter ces requêtes plus complexes.



Hashkey	Objects	hasChildren
$\tilde{h}^2(0,0)$	A	false
$\tilde{h}^2(4,0)$	A	true
$\tilde{h}^2(0,4)$	A	false
$\tilde{h}^2(4,4)$	A	false
$\tilde{h}^1(4,0)$		true
$\tilde{h}^1(6,0)$		true
$\tilde{h}^0(5,1)$	B	
$\tilde{h}^0(5,0)$	B	
$\tilde{h}^0(6,1)$	B	
$\tilde{h}^0(6,0)$	B	

Point P
$\downarrow$
$\tilde{h}^2(4,0)$: retrieve object A
$\downarrow$
$\tilde{h}^1(4,0)$
$\downarrow$
$\tilde{h}^0(5,O)$: retrieve object B

Point Q
$\downarrow$
$\tilde{h}^2(0,4)$: retrieve object A

Figure 2.3 – Le schéma montre le hachage d'objets mobiles. La liste en bas à gauche représente la table de hachage obtenue avec notre méthode. Le schéma en bas à droite représente la phase de détection de collisions pour les points P et Q.

Algorithme 2.2 Obtention des objets en collision avec une boite englobante.

```
set<int> broadPhase(AABB a) {
 set<int> candidateObjects;
 // Build a set with the 8 endpoints
 // (the initial hash key for these points is set to 0)
 vector<point> points = a.getEndPoints();

 for each (int currSize in classSizes) {
  int delta = a.size() - currSize;
  // Subdivide AABB if necessary
  if (delta > 0) points.insert(a.subdivide(delta));
  // Find possible candidates in corresponding cells
  for each (point p in points) {
   p.hashKey += h(p, currSize);
   candidateObjects.insert(hm[p.hashKey].retrieveIds());
   // Remove this point if the cell has no child
   if (! hm[p.hashKey].hasChildren) points.remove(p);
  }
 }
 return candidateObjects;
}
```

2.2 Détection des collisions

La détection des collisions consiste, dans la phase approximative, à éliminer le plus d'objets possible pour la phase exacte. Le critère d'élimination est généralement un critère spatial : la distance entre les boites englobantes des objets permet de ne garder que les objets dont les boites englobantes se chevauchent, et qui sont donc potentiellement en collision. La vérification de la collision exacte étant une opération coûteuse, il est impératif de l'utiliser le moins possible. Il est donc crucial que le nombre de candidats obtenus lors de la première phase soit le plus petit possible.

Nous nous intéressons maintenant à vérifier si une boite englobante donnée chevauche des cellules dans la structure de hachage qui peuvent contenir des candidats potentiels pour la détection de collisions exacte (cette procédure est décrite par l'Algorithme 2.2). Dans le contexte des interactions fluide-solide, la boite englobante considérée est celle du parcours de la particule. Dans un contexte plus général, la boite englobante correspond aux objets statiques ou mobiles qui sont potentiellement en collision avec d'autres objets de la scène (Figure 2.3). Le principal avantage de notre méthode est l'utilisation du booléen *hasChildren* qui permet de terminer rapidement le processus de recherche dans les zones vides. Comme la boite englobante a considérée est de taille arbitraire, il peut être nécessaire de la subdiviser en boite plus petites en utilisant *a.subdivide()* d'une façon similaire à la décomposition d'un

Algorithme 2.3 Insertion d'une particules et récupération des particules voisines.

```
void insert(particle P) {
 set<int> candidateParticles;
 // Build a set with P and 26 neighbouring points
 // (the initial hash key for these points is set to 0)
 vector<point> points = get27Points(P.position);

 for each (int currSize in classSizes)
  // Retrieve possible candidates if size r is reached
  for each (point p in points) {
   p.hashKey += h(p, currSize);
   if (currSize == r)
    candidateParticles.insert(hm[p.hashKey].retrieveIds());
   if (! hm[p.hashKey].hasChildren) points.remove(p);
  }
 // Update the 'neighbours' attribute if the distance
 // between particles is lower than r
 for each (int id in candidateParticles)
  if (distance(P.id, id) <= r) {
   neighbours[P.id].insert(id);
   neighbours[id].insert(P.id);
  }
 // Insert the particle (finally !)
 insert(P.position, P.id, r);
 }
}
```

octree afin de s'assurer de ne manquer aucune cellule plus petite que *a.size()*. Cette opération est par exemple nécessaire quand le parcours d'une particule est plus grand que les plus petits objets de la scène. Une utilisation plus complexe peut optimiser ce processus en s'assurant qu'une valeur *p.hashkey* n'est pas traitée plusieurs fois ce qui peut se produire si *a.size()* est très petit et plusieurs points de la boite englobante peuvent être hachés dans les mêmes cellules.

2.3 Recherche de voisinage

Comme indiqué précédemment, une particule p insérée dans une cellule doit chercher ses particules voisines situées dans son rayon d'interaction r en cherchant dans la même cellule et dans ses 26 voisines. Les clés de hachage de ces cellules voisines sont simplement obtenues par : $\widetilde{h}^r(p_x + r, p_y + r, p_z + r)$, $\widetilde{h}^r(p_x - r, p_y + r, p_z + r)$, $\widetilde{h}^r(p_x - r, p_y - r, p_z + r)$, ...

Algorithme 2.4 Obtention des points dans les cellules voisines.

```
vector<point> get27Points(point p) {
 vector<point> result;
 result.push_back(p);
 for (float x = p.x - r; x < p.x + r; x += r)
  for (float y = p.y - r; y < p.y + r; y += r)
   for (float z = p.z - r; z < p.z + r; z += r)
    result.push_back(point(x,y,z));
 return result;
}
```

Cependant, comme r est la plus petite classe, cette approche simple n'est pas très efficace avec notre méthode puisque le calcul des clés de hachage nécessite de prendre en compte toutes les classes de *classSizes*. Une solution est de déterminer les particules voisines directement lors de l'insertion de chaque particule. Chaque particule dispose donc d'une liste de références des particules voisines mises à jour à mesure que les particules sont insérées (Algorithme 2.3). Dans un premier temps les points correspondant aux cellules de classe r voisines de la particule P sont calculés avec la fonction *get27Points* (Listing 2.4). Puis la distance des particules déjà présentes dans ces cellules est évaluée pour chacun des 27 points avant l'insertion de la particule. Cette méthode permet de diminuer les calculs de collisions entre particules car ils ne sont effectués qu'une seule fois par couple de particules, la relation de voisinage étant calculée par les particules insérées après leurs voisines. En effet, un des éléments les plus coûteux dans la recherche de voisinage est le calcul de collision entre particules. Le deuxième avantage pour les premières particules insérées dans la structure est qu'elles n'ont pas encore de particules voisines et donc les variables booléennes des cellules parentes indiquent que les cellules filles sont vides. La plupart des 27 points calculés pour les premières particules sont donc éliminés rapidement dans les premières classes de hachage.

2.4 Résultats et comparaisons

Les résultats présentés dans cette section comparent les performances de notre méthode avec d'autres structures de données sur des distribution de tests aléatoires en 2D et avec une simulation de fluide en 3D. La machine utilisée pour ces tests dispose d'un processeur i686 à 2.6GHz.

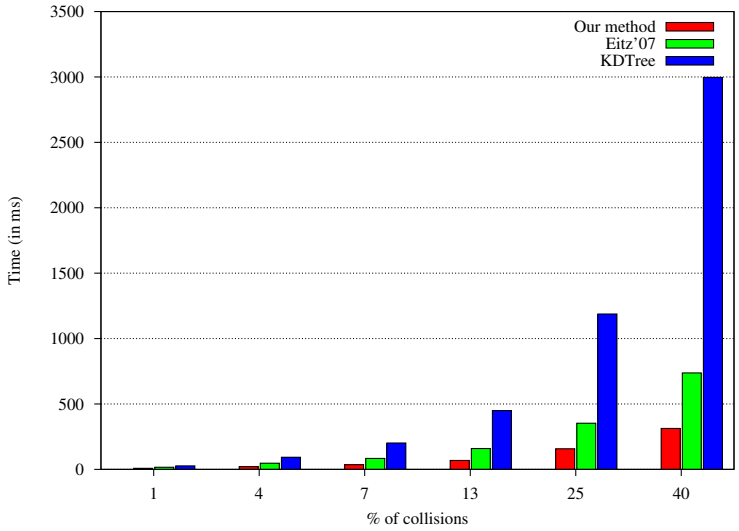


Figure 2.4 – Comparaison des temps de calcul pour la phase de détection de collisions approximative.

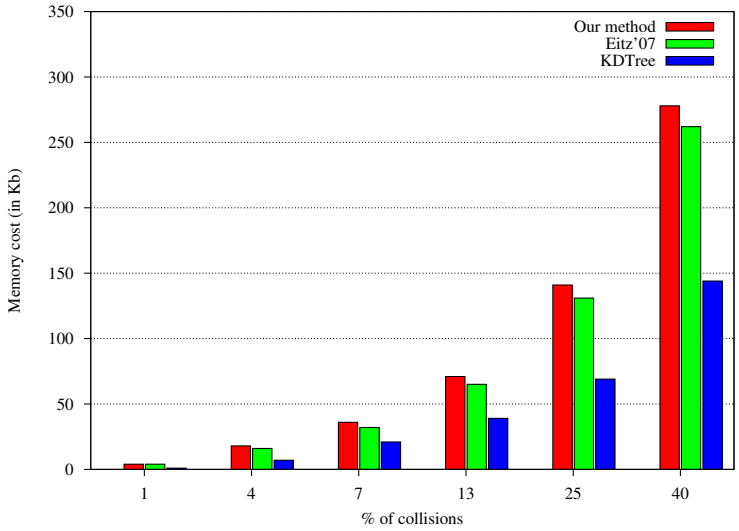


Figure 2.5 – Comparaison de la consommation mémoire.

2.4.1 Comparaison sur une distribution aléatoire 2D

Les Figures 2.4 et 2.5 montrent les résultats des tests de la phase de détection de collision approximative. Notre structure de données est comparée avec le KD-tree et le schéma de hachage multi-niveaux de Eitz et Lixu [EL07]. Une distribution de N objets est générée aléatoirement dans un carré de taille 2^{13}, puis insérée dans la structure d'accélération en utilisant la boite englobante de chaque objet. Ensuite une distribution de $1M$ de segments représentant les particules en déplacement dans la scène est calculée. Chacun des segments effectue une requête dans la structure pour déterminer quels objets sont potentiellement intersectés par ces segments. La répartition de la taille de segment est choisie de sorte qu'environ 10% des objets aient une taille moyenne de 2^{10}, 25% aient une taille moyenne de 2^6 et les objets restant aient une taille moyenne de 2^4. La taille du vecteur de déplacement de chacune des particules ne dépasse pas 2^4. Les temps de calcul sont représentés en fonction du ratio de particules en collision avec au moins N objets, ce ratio augmente selon cette valeur N. Notre méthode permet d'améliorer les temps de calculs d'au moins un facteur deux par rapport à la méthode décrite dans [EL07] dans toutes les situations, alors que la quantité de mémoire supplémentaire nécessaire ne dépasse pas 10%. Nos tests indiquent de plus que les structures basées sur une méthode de hachage sont en général plus efficaces que le KD-tree pour la phase de détection de collisions approximative, bien qu'elles nécessitent plus de mémoire.

Nous avons aussi évalué le potentiel de notre structure de données par rapport au KD-tree et à une grille uniforme dans le cas des opérations de recherche de voisinage. Un ensemble de $1M$ de points représentant des particules est distribué aléatoirement dans un carré de taille 2^{20} et inséré dans la structure d'accélération, ensuite chaque point recherche dans la structure l'ensemble des points voisins situés à une distance inférieure à r. La Figure 2.6 représente les temps de calcul obtenus en fonction du nombre de voisins contenus dans un rayon croissant r et la Figure 2.7 représente la consommation de mémoire pour le même nombre de voisins. Nous remarquons que notre structure est compétitive par rapport à une grille uniforme si le nombre de voisins ne dépasse pas 5, c'est-à-dire si les points sont assez éloignés les uns des autres. Cependant, la quantité de mémoire supplémentaire nécessaire est comprise entre 10% et 25%. Dans le cas d'une distribution plus dense de points, le KD-tree est la structure la plus performante, ce qui est partiellement du au fait que dans le cas de la recherche de voisinage, tous les objets sont de la même taille. Les méthodes de simulation utilisant des particules possédant un nombre de voisines élevé devraient donc préférer un KD-tree aux structures de hachage.

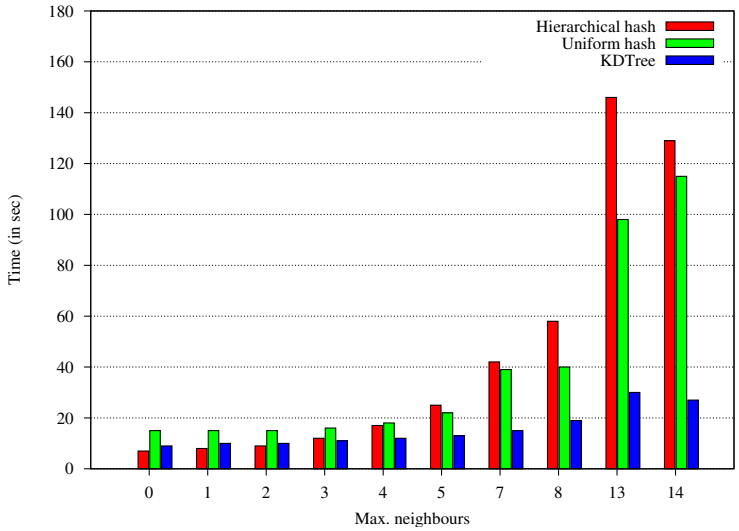


Figure 2.6 – Comparaison des temps de calculs pour la recherche des particules voisines.

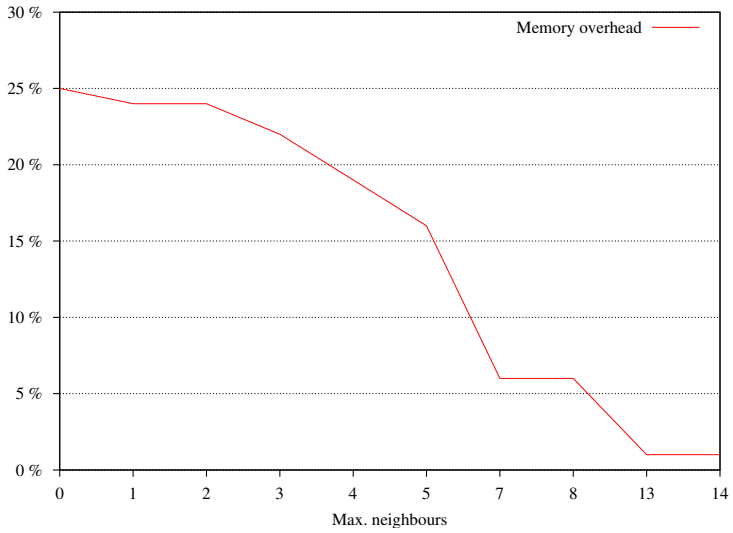


Figure 2.7 – Mémoire supplémentaire nécessaire pour notre méthode.

2.4.2 Comparaison sur une simulation de fluide 3D

La comparaison des structures de données consiste à mettre en collision les particules d'un fluide avec 2000 sphères immobiles. Les particules du fluide finissent par se stabiliser et atteindre une position d'équilibre au fond du domaine de simulation durant 1000 itérations. Nous avons retenu comme modèle de fluide la méthode DEM, reposant sur la production de simples forces de réponse aux collisions des particules. D'autres méthodes lagrangiennes comme la méthode SPH [MCG03] peuvent être utilisées. Les particules mobiles du fluide et les sphères immobiles sont placées aléatoirement dans le domaine de simulation. La taille des sphères varie de 2^{-4} à 2^{2}, alors que les particules possèdent une taille de 2^{-6} (2.8).

Les performances correspondent au temps requis pour effectuer l'ensemble du processus de détection de collision pour chaque itération de la simulation. Le processus de détection de collision comprend plusieurs étapes : insérer les objets dans la structure d'accélération (même si les sphères et les murs sont statiques dans cet exemple), insérer les particules mobiles, récupérer l'ensemble des candidats de chaque particule et finalement calculer les collisions exactes avec les candidats. La recherche de voisinage est toutefois effectuée avec un hachage uniforme. Comme le montre la Figure 2.9, notre méthode surpasse celle de Eitz et Lixu d'environ 25%. La mesure de la consommation mémoire (non présentée ici) montre que la mémoire supplémentaire nécessaire pour stocker les informations de relations hiérarchiques ne dépasse jamais 20%, comme dans le cas 2D. Il est important de remarquer que l'agrégation progressive des particules au fond de la scène au cours de la simulation diminue le nombre de particules isolées. Ceci explique pourquoi l'accélération obtenue par notre méthode n'est pas aussi spectaculaire que dans le cas 2D mais reste malgré tout importante. Cette structure est donc encore plus efficace dans le cas de la simulation d'un fluide turbulent.

2.5 Implémentation et discussion

Les programme de test 2D dont les résultats sont présentés dans ce chapitre ont été implémentés avec le conteneur *hash_multimap* fournit par la bibliothèque STL du C++. Cette implémentation naïve permet de ne pas utiliser de taille interne fixe pour la structure de hachage. Ceci implique que l'opération modulo présente dans les équations 2.2 et 2.1 n'est plus nécessaire et que l'on peut considérer $n = +\infty$. Cependant cette solution n'est pas idéale pour autant. En effet, la structure de données utilisée par ce conteneur possède une taille fixe mais qui est augmentée à mesure que des collisions de hachage apparaissent. La clé de hachage des éléments insérés dans la table est alors recalculée, ce qui explique les performance de notre structure face au KD-tree lors de l'utilisation d'un nombre élevé d'objets. Cette structure de données

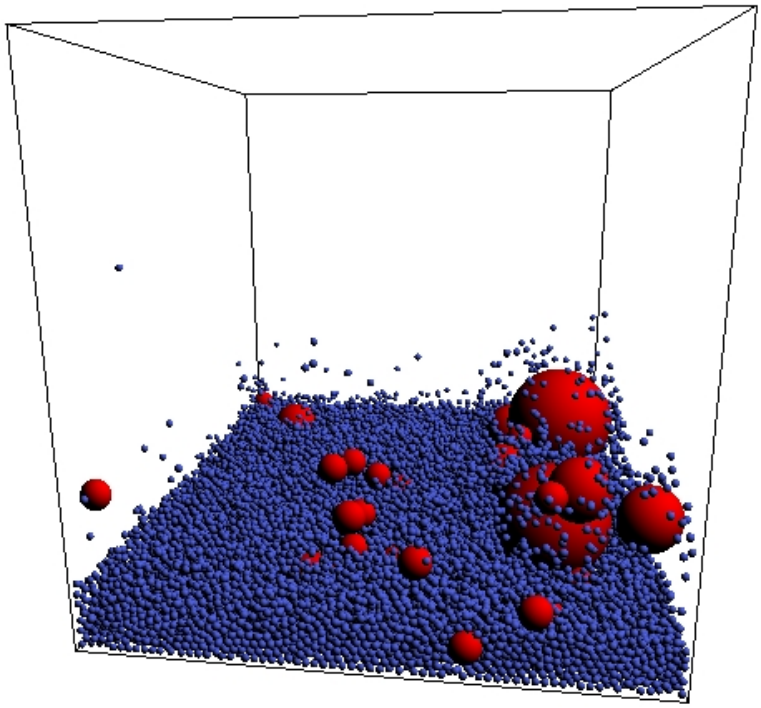


Figure 2.8 – Notre scène de test 3D

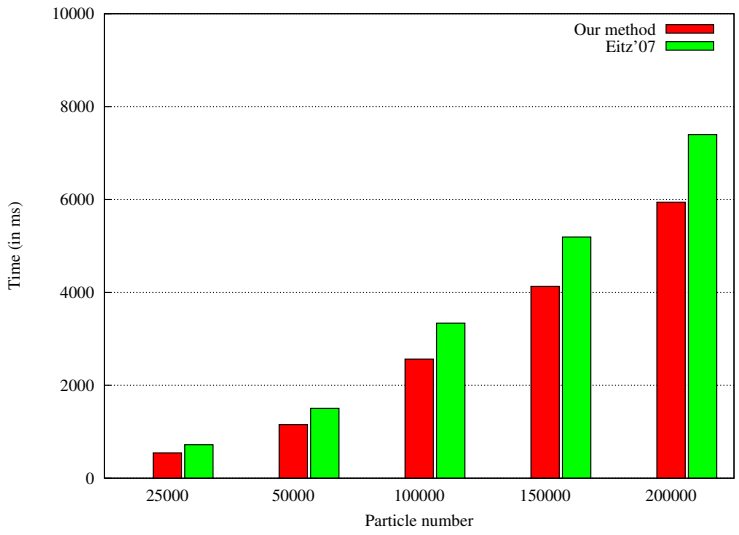


Figure 2.9 – Comparaison des performances de la détection de collisions (en milliseconde par étape de la simulation).

permet donc d'obtenir des performances intéressantes avec $1M$ de particules tout en laissant de bonnes possibilités d'optimisations.

Dans le cas des tests 3D, une version plus optimisée a été implémentée, basée sur l'utilisation d'un tableau de taille fixe. Nous avons de nouveau obtenu des résultats satisfaisants par rapport à l'approche de Eitz et Lixu, mais dans ce cas la valeur de la taille de la table n doit être choisie pour toute la simulation. Le choix de cette valeur a un impact important sur la structure de données : si n est trop grand la quantité de mémoire nécessaire sera importante, alors qu'une valeur trop petite augmentera les collisions de hachage et donc le nombre d'objets candidats pour la phase exacte de détection de collisions. Nos expériences montrent que le coût des calculs supplémentaires induit par les « faux » candidats est négligeable quand n est fixé au nombre moyen de polygones présents dans la scène multiplié par 100.

Dans le cas précis des simulations dynamiques, il est possible d'utiliser un marqueur temporel, c'est-à-dire un paramètre qui distingue à quelle itérations appartient un objet inséré dans la structure [THM+03]. Lors de l'insertion d'un objet dans une cellule de hachage, si le marqueur des objets présents dans cette cellule diffère de celui de l'objet à insérer, alors il n'est pas nécessaire d'allouer un nouvel espace mémoire, il suffit de réutiliser la mémoire allouée pour un des autres objets. Ce marqueur permet de réduire le total des allocations mémoire lors de la mise à jour de la structure. En utilisant ce comportement la structure n'est jamais totalement effacée entre chaque itération successive de la simulation. Une implémentation basée sur la bibliothèque STL permet d'obtenir ce fonctionnement car l'opération *clear* ne libère pas nécessairement la mémoire quand elle est utilisée à chaque itération.

2.6 Conclusion

L'utilisation de structures de données adaptées est un des facteurs déterminant les performances d'une simulation de fluide. En général les méthodes basées sur l'utilisation d'une structure de hachage permettent d'obtenir de meilleures performances que le KD-tree pour la phase de détection de collisions. En particulier, l'utilisation d'un schéma de hachage hiérarchique permet de réduire fortement les calculs dans les régions vides de la simulation et permet donc d'améliorer encore les performances par rapport au KD-tree mais aussi par rapport au hachage hiérarchique de Eitz et Lixu.

Les structures de hachage sont aussi intéressantes dans le cas de la recherche de voisinage, et le schéma de hachage hiérarchique est performant si les points sont assez éloignés entre eux, ce qui est le cas pour un fluide turbulent par exemple. Dans le cas d'une distribution plus dense de points, le KD-tree est une structure plus intéressante. En effet la recherche de voisinage ne bénéficie pas de la cohérence hiérarchique dans

cette configuration de particules. En pratique il est souvent préférable d'utiliser une structure différente pour la recherche de voisinage.

Le cas de la simulation d'un fluide composé de particules de tailles différentes, ce qui est le cas des méthodes adaptatives, est toutefois différent et les structures de hachage hiérarchiques semblent plus intéressantes que le KD-tree dans cette situation. Nous nous intéresserons en particulier dans le Chapitre 3 à l'utilisation de structures hiérarchiques pour la recherche de voisinage dans le cas de la simulation de fluide adaptative.

Chapitre 3

Recherche de voisinage pour les simulations de fluide lagrangiennes adaptatives

La partie critique des méthodes de simulation de fluide lagrangiennes est la phase de recherche de voisinage qui repose généralement sur l'utilisation d'une structure de données performante. Cette partie est encore plus critique dans le cas des méthodes adaptatives qui peuvent contenir des particules de tailles très différentes. Les structures de données employées habituellement, comme les grilles uniformes ou tables de hachage ne sont pas adaptées pour traiter des ensembles de particules de tailles variables puisqu'elles utilisent des cellules de taille fixe. Dans le cas d'une méthode multi-résolution, chaque particule doit étendre son rayon de recherche et le nombre de cellules visitées ce qui diminue fortement les performances de la structure. Les structures arborescentes comme le KD-tree sont plus adaptées puisqu'elles ne reposent pas sur l'utilisation de cellules de taille fixe. Ces structures souffrent cependant d'un temps de construction trop long pour être reconstruites à chaque pas de temps de la simulation. Enfin, les tables de hachage et grilles hiérarchiques sont utilisées de manière judicieuse pour la détection de collision ou dans le cas des interactions fluide-solide (présenté dans le Chapitre 2). Ces structures permettent de traiter efficacement des objets de tailles très différentes et leur utilisation semble donc pertinente pour la simulation multi-résolution. Toutefois la recherche de voisinage exige une utilisation adaptée de ces structures.

Les méthodes adaptatives ont aussi besoin d'évaluer les critères leur permettant d'adapter la taille des particules. L'évaluation de ces critères est souvent une opération coûteuse car nécessitant une connaissance globale de la répartition des particules dans la scène. L'utilisation de la grille hiérarchique permet d'utiliser des critères locaux évalués à des niveaux différentes de subdivision.

Enfin, la phase de rendu est aussi une étape coûteuse de la simulation et le choix judicieux des particules utilisées pour cette étape est donc important. En effet, il n'est pas nécessaire de prendre en compte l'ensemble des particules composant le fluide, en particulier les particules situées à l'intérieur du fluide sont inutiles puisque non visibles de la surface.

Dans la suite de ce chapitre nous verrons comment il est possible d'utiliser les structures de tables de hachage et grilles hiérarchiques pour l'étape de recherche de voisinage dans le cas d'un fluide multi-résolution (3.1). Puis nous montrerons une méthode de simulation adaptative utilisant les données calculées dans la grille hiérarchique comme critère de fusion / subdivision des particules. Les travaux présentés dans ce chapitre ont fait l'objet d'une communication dans la conférence *Computer Animation and Social Agents (Short Papers Session)* [PCG10].

3.1 Grilles hiérarchiques pour la recherche de voisinage

Les grilles hiérarchiques partitionnent l'espace en une hiérarchie de cellules cubiques d'une façon similaire à la décomposition d'un octree mais sans qu'il soit nécessaire de stocker les liens entre les cellules filles et mères. En effet la position des cellules filles ou mères peut être facilement calculée en fonction de la cellule courante.

3.1.1 Construction de la grille hiérarchique

Dans un premier temps il est nécessaire de construire les niveaux de la grille en fonction des tailles de particules qui seront présentes dans la simulation. En pratique le premier niveau l_1 est choisi afin de pouvoir contenir les plus petites particules de la simulation. Les niveaux suivants $l_2, l_3, ..., l_n$ sont alors construits jusqu'au niveau pouvant contenir les plus grandes particules de la simulation. Pour chaque niveau, la taille d'une cellule est une puissance de 2, la taille est donc doublée par rapport à la taille d'une cellule du niveau inférieur. Il est intéressant de calculer avant de commencer la simulation, pour une gamme de différents rayons de particules, les niveaux qui sont associés à ces rayons. Cette optimisation est avantageuse dans le cas où les différentes tailles de particules qui seront présentes dans la simulation sont connues par avance.

3.1.2 Utilisation de la grille hiérarchique

Le but est maintenant d'insérer chaque particule de la simulation dans le niveau l de la grille le plus approprié en fonction du rayon r de la particule. Le niveau le plus approprié pour une particule est le premier niveau dans lequel une cellule peut contenir entièrement cette particule tel que :

$$2^l \geq 2r > 2^{l-1} \tag{3.1}$$

Ce niveau est donc obtenu en calculant la première puissance de 2 plus grande que le diamètre $2r$ de la particule avec l'équation :

$$l = \lceil log_2(2r) \rceil \tag{3.2}$$

et la taille de cellule k associée à ce niveau est donc :

$$k = 2^l \tag{3.3}$$

Une fois que le niveau correspondant à une particule est déterminé, l'insertion de la particule dans ce niveau est similaire à l'insertion d'une particule dans une grille uniforme. La particule est simplement insérée dans la cellule dont le centre est le plus proche du centre de la particule, les coordonnées entières de la cellule sont calculées avec l'équation 1.23. La Figure 3.1 présente un exemple de grille hiérarchique et des particules associées à chaque niveau. Il est important de noter que si une taille de particule est associée à un seul niveau de la grille, chaque niveau peut contenir des particules de plusieurs tailles.

Une fois les particules insérées dans la structure, la recherche de voisinage est presque aussi simple que dans le cas d'une grille uniforme. Dans un premier temps une particule commence par chercher ses voisines dans les 27 cellules voisines de son niveau. Puis elle effectue la même recherche de façon ascendante pour tous les niveaux supérieurs à son niveau jusqu'au niveau le plus haut de la grille (Figure 3.2). Il n'est pas nécessaire pour une particule de rechercher ses voisines dans les niveaux inférieurs à son niveau puisque la relation de voisinage entre deux particules de tailles différentes est calculée par la particule la plus petite. De plus, le rayon des cellules vérifiées par la particule devrait alors être étendu et réduirait fortement les performance de la recherche de la même manière que dans le cas de la recherche dans une grille uniforme.

L'Algorithme 3.1 présente la méthode de recherche de voisinage qui permet d'obtenir une liste de particules candidates pour un test de collision plus précis. Ce test est lui même présenté dans l'Algorithme 3.2 nécessaire pour le calcul des forces résultant des éventuelles collisions entre les particules.

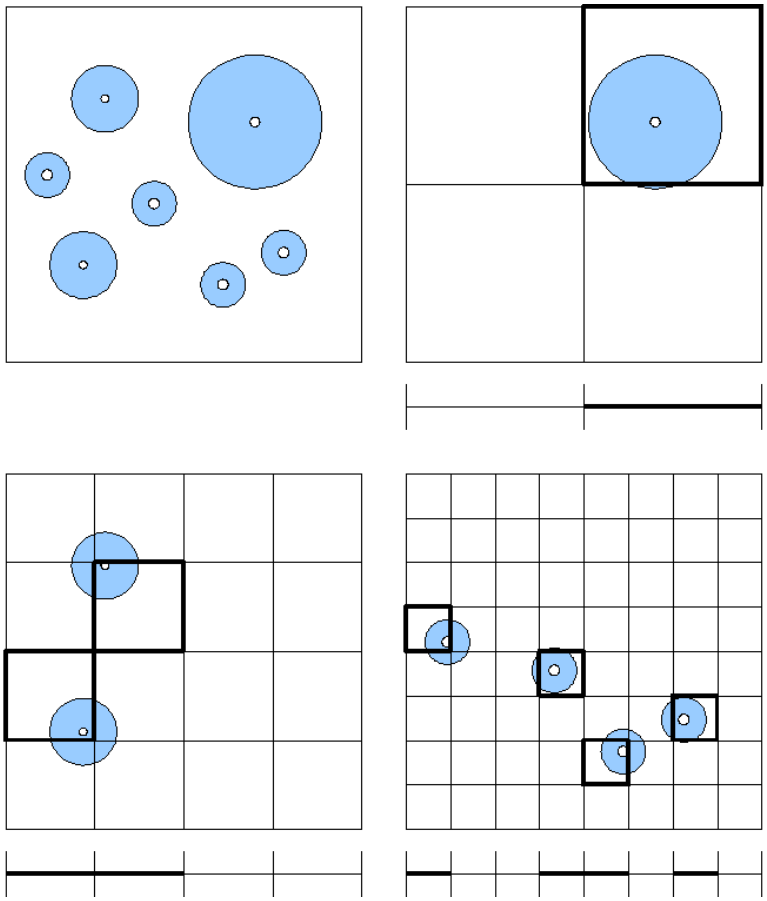


Figure 3.1 – Dans cet échantillon chaque particule est insérée dans le niveau de la grille approprié en fonction de son rayon. Les trois tailles de rayon différentes produisent trois niveaux dans la grille hiérarchique qui sont présentés en 2D et 1D. Les cellules ayant des traits gras sont celles contenant les particules.

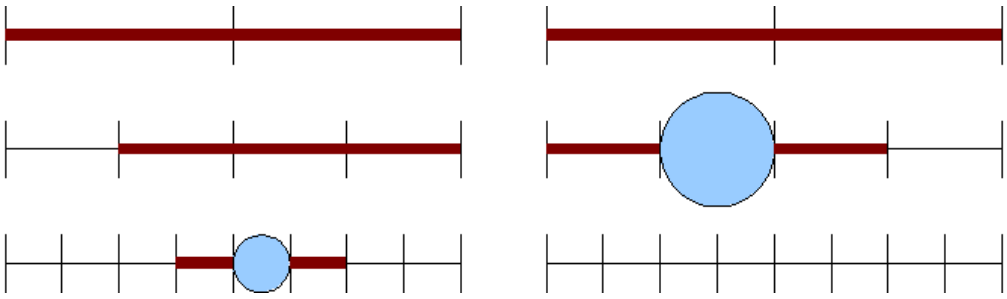


Figure 3.2 – Représentation 1D de la recherche de voisinage pour des particules dans une distribution aléatoire, pour la plus petite taille de la simulation (gauche) et une taille intermédiaire (droite). Les particules candidates sont recherchées seulement dans le niveau de la particule et dans les niveaux supérieurs.

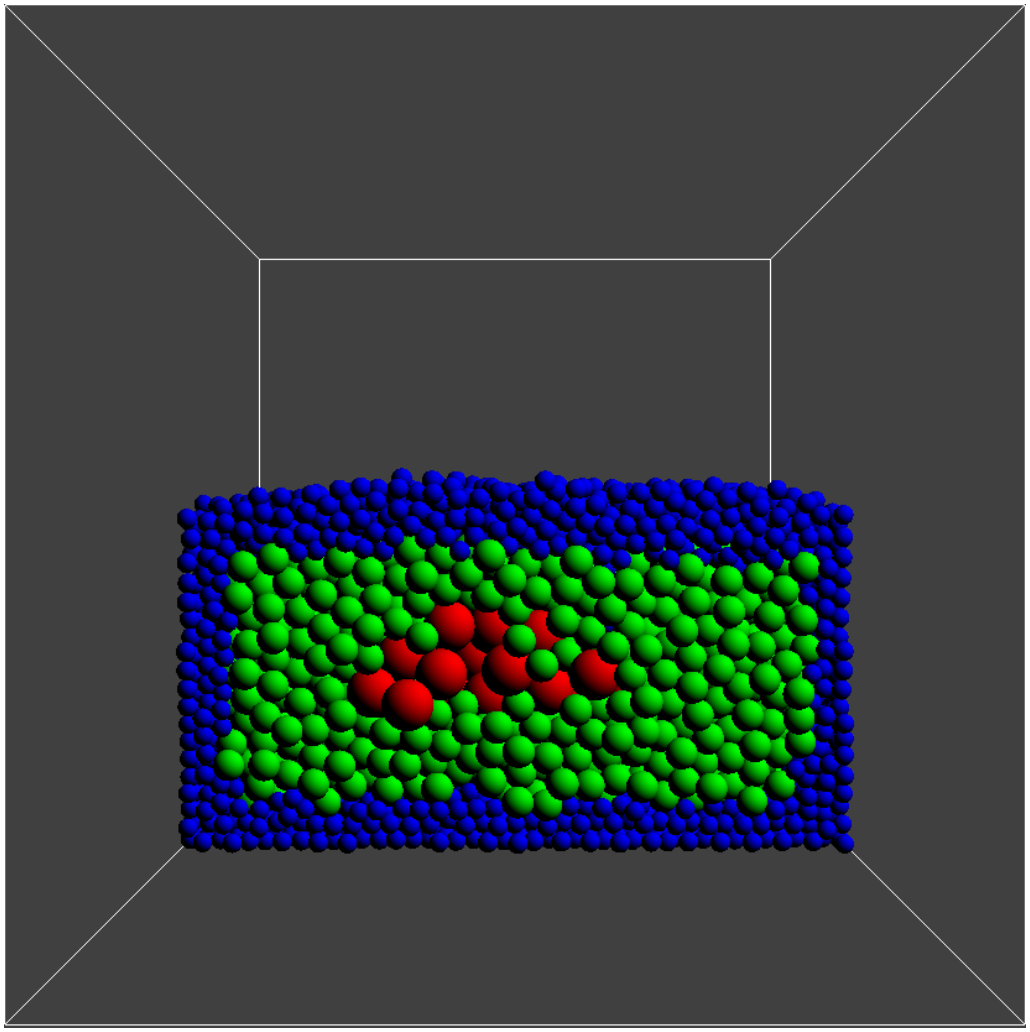


Figure 3.3 – Exemple de répartition des particules du fluide en couches de trois tailles différentes.

3.1.3 Recherche de voisinage pour une distribution de particules optimisée

Le schéma d'utilisation de la grille hiérarchique présenté dans la Section 3.1.2 permet à des particules de tailles différentes de rechercher leurs voisines dans une distribution de particules aléatoire, c'est-à-dire une distribution dans laquelle des particules de tailles très différentes sont totalement mélangées. Dans cette situation, les particules n'ont pas de connaissance a priori des particules qui les entourent. Ce type de distribution a plusieurs inconvénients :

- Il est nécessaire pour chaque particule de vérifier l'ensemble des niveaux supérieurs au sien afin d'obtenir l'ensemble des particules potentiellement en collision avec elle.
- Les grandes particules entourées par des particules beaucoup plus petites gé-

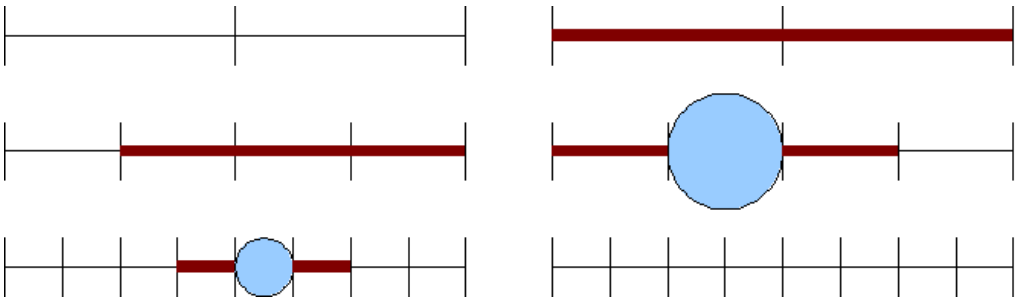


Figure 3.4 – Représentation 1D de la recherche de voisinage pour des particules dans une distribution optimisée, pour la plus petite taille de la simulation (gauche) et une taille intermédiaire (droite). Les particules candidates sont recherchées seulement dans le niveau de la particule et dans le niveau supérieur.

nèrent plus de contacts entre les particules que dans une simulation de particules de taille unique et augmente donc le nombre de calculs de collisions.

Les méthodes adaptatives cherchent au contraire à grouper les particules en fonction de leurs tailles [APKG07, YWH+09, HHK08, HHK09, ZSP08], en maintenant une distribution de particules sous forme de couches dans lesquelles les particules ne peuvent être en contact qu'avec des particules de même niveau ou ayant au plus un niveau d'écart (Figure 3.3). L'utilisation d'une distribution de particules sous forme de couches permet ainsi de minimiser les contacts entre particules de différentes tailles.

Pour ce type de distributions le schéma d'utilisation de la grille hiérarchique peut être optimisé. En effet, les particules doivent rechercher leurs voisines dans au plus deux niveaux : leur niveau et le niveau directement supérieur comme le montre la Figure 3.4.

3.1.4 Extension des grilles hiérarchiques en tables de hachage

Dans certaines situations les besoins en mémoire peuvent être plus importants que les performances brutes. En effet, la structure de grille hiérarchique nécessite d'avoir une zone mémoire pour chaque niveau de la grille. De plus, le domaine de simulation représenté par la grille hiérarchique est borné, ce qui peut être une limitation dans le cas d'un fluide très mobile. Dans ces situations il est possible d'utiliser une table de hachage hiérarchique permettant de stocker toutes les particules dans une zone mémoire unique de taille adaptée. Comme le décrivent les Chapitres 1 et 2, l'utilisation d'une table de hachage permet de simuler le déplacement des particules sur un domaine virtuellement infini. De plus tous les niveaux dans la structure de hachage sont stockés dans la même zone mémoire, il est donc possible d'utiliser autant de niveaux que nécessaire. Cependant, les tables de hachages sont soumises aux collisions de hachage qui augmentent le nombre de candidats potentiels et diminuent donc les performances de la recherche de voisinage.

Algorithme 3.1 Recherche des particules voisines.

```
set<particle*> getCandidateParticles(particle p) {
 // List of potential colliding particles
 set<particle*> candidates;
 point pos = p.getPosition();

 // Relevant levels are chosen according to
 // optimal or random particle distribution
 for each (int level in relevantLevels) {
  index originCell = computeGridCell(pos,level);
  for(int z = -1; z <= 1; z++)
   for(int y = -1; y <= 1; y++)
    for(int x = -1; x <= 1; x++) {
     // Compute corresponding neighboring grid cell
     index gridCell = originCell + index(x, y, z);
     // Get all particles stored in this cell
     getCandidateFromCell(gridCell,candidates, level);
    }
 }
 return candidates;
}
```

Algorithme 3.2 Calcul de la force résultant des collisions entre particules.

```
void computeResponseForce(particle p,
                          set<particle*> candidates) {
 // Initialize particle's force
 p.force = vector(0, 0, 0);
 for each (particle c in candidates) {
  if (dist(p, c) < (p.radius + c.radius)) {
   // Particles are colliding compute response force from
   // position and velocity
   vector force = computeCollisionResponse(p, c);
   p.force += force;
   if (p.radius < c.radius)
    // Handle response force for bigger particles
    c.force -= force;
  }
 }
}
```

Nombre de fusions	Rayon	Taille des cellules
0	1	$2 \cdot 2^0 = 2 \cdot 1$
1	$1.7874...$	$2 \cdot 2^1 = 2 \cdot 2$
2	$2.5198...$	$2 \cdot 2^2 = 2 \cdot 4$
...	...	...

Table 3.1 – Le nombre de fusions représente le nombre de fois qu'une particule a été fusionnée. Le rayon de la nouvelle particule est calculé avec l'équation 3.4 et la taille de cellule adaptée est calculée avec l'équation 3.3.

3.2 Grilles hiérarchiques pour la simulation adaptative

Notre objectif est maintenant d'utiliser la structure de grilles hiérarchiques non seulement pour la recherche de voisinage mais aussi comme un moyen de déterminer si une particule doit être fusionnée ou subdivisée en utilisant des critères simples. De plus, nous utilisons cette structure hiérarchique afin d'identifier les particules les plus pertinentes pour le rendu du fluide. En effet, dans la plupart des cas il n'est pas utile de considérer les particules situées à l'intérieur du fluide puisqu'elle ne sont pas visibles depuis la surface. Les particules de la surface sont principalement responsables de la forme du fluide. Afin d'identifier ces particules nous voulons aussi utiliser une méthode simple basée sur la structure hiérarchique. Pour ces deux problèmes nous utilisons une méthode similaire mais à des niveaux différents de la grille.

Dans la méthode que nous décrivons nous faisons la distinction entre les *particules de surface* et les *particules de bordure*. Les particules de surface sont les particules en contact avec les objets de la scène ou directement situées à la surface du fluide. Les particules de bordure sont proches des objets ou de la surface du fluide et représentent une description large de la surface du fluide. Les particules de surface sont alors évidemment aussi des particules de bordure. Enfin, les particules qui ne sont pas des particules de bordure sont situées à *l'intérieur* du fluide. Afin d'identifier les particules de surface ou de bordure nous nous appuyons uniquement sur des calculs locaux utilisant la grille hiérarchique.

3.2.1 Schéma de fusion / subdivision adaptatif

Nous avons choisi d'utiliser un schéma adaptatif simple consistant à fusionner et subdiviser les particules en fonction de critères locaux. Dans notre schéma adaptatif les plus grandes particules sont contraintes à évoluer à l'intérieur du fluide et les plus petites particules sont situées à la surface du fluide ou près des obstacles. Ainsi le fluide peut conserver une surface détaillée et des collisions plus précises avec son environnement. La partie intérieure ayant beaucoup moins d'influence sur l'écoulement du fluide, il est donc judicieux d'approximer cette partie du fluide en fusionnant les

particules. Pour réaliser cet objectif, il est nécessaire d'organiser les particules sous forme de couches comme le décrit la Section 3.1.3. Chaque couche contient des particules d'une même taille qui doivent fusionner ou se subdiviser si elles sortent de leur couche à cause de l'écoulement du fluide afin de respecter les contraintes énoncées précédemment. En pratique, il est difficile de fusionner les particules de telle sorte qu'elles forment directement des couches uniformes en utilisant des critères locaux. Dans notre cas, des particules de deux tailles différentes peuvent être mélangées jusqu'à la formation complète d'une couche.

Nous avons choisi un schéma de fusion / subdivision simple dans lequel 4 particules de même taille peuvent fusionner en une particule de taille supérieur et une particule peut se subdiviser en 4 particules plus petites de même taille. Le rayon r_2 d'une nouvelle particule résultant d'une fusion de 4 particules ayant pour rayon r_1, peut être calculé par :

$$r_2 = r_1 \cdot \sqrt[3]{4} \tag{3.4}$$

En utilisant ce schéma, chacun des trois premiers niveaux de la grille l_1, l_2, l_3 est associé à un seul rayon de particule r_1, r_2, r_3 ce qui est pratique pour notre schéma adaptatif puisque nous pouvons utiliser chaque niveau pour estimer la densité de particule d'une taille précise (Table 3.1). Comme le nombre maximum de fusion d'une particule est connu il est alors possible de calculer le rayon associé à chaque niveau de fusion. Même si nous n'utilisons que trois niveaux de particules dans notre implémentation, la grille hiérarchique est capable de traiter des particules de tailles quelconques. Le choix de la fusion des particules par groupe de quatre est aussi un compromis par rapport aux erreurs produites par le processus de fusion. En effet, plus le nombre de particules utilisées pour une fusion est grand, plus les erreurs sont importantes et le résultat visible sur le fluide sous la forme de remous dus à l'augmentation des forces de pression après une fusion. De plus, il peut être difficile de trouver suffisamment de particules voisines pour réaliser l'opération de fusion.

La partie critique est maintenant d'identifier l'état de chaque particule, qui peut être de trois types différents, sans avoir une connaissance globale de la répartition des particules :

- Une particule à l'intérieur du fluide devrait fusionner.
- Une particule près de la surface devrait se subdiviser.
- Une particule qui n'est dans aucun de ces deux états n'a aucune opération à effectuer.

Ces états définissent les critères que nous utilisons dans notre simulation et correspondent aux définitions énoncées précédemment des particules intérieures, de surface et de bordure, respectivement. Nous étendons la définition de ces particules à tous les niveaux de la grille afin d'appliquer ces critères sur chacune des couches de particules.

De cette manière, chaque couche peut garder une frontière de particules permettant de séparer les particules ayant deux niveaux d'écart. Les particules qui doivent fusionner ou se subdiviser sont donc choisies en utilisant les critères suivants :

- Les seules particules qui sont fusionnées sont les particules qui n'ont pas de particules de bordure dans leur voisinage et qui ne sont pas elles mêmes des particules de bordure. Ce critère assure que la fusion de particules se produit à une distance suffisante de la surface.
- Les particules sont subdivisées lorsqu'elles sont des particules de surface qui n'ont pas de particules plus grandes qu'elles dans leur voisinage. Les particules qui sont déjà de la plus petite taille de la simulation ne doivent évidemment pas être subdivisées. Ce critère assure que les particules près de la surface aient la plus petite taille possible et empêche les grandes particules de s'approcher de particules beaucoup plus petites puisqu'elles doivent se subdiviser avant d'atteindre leur voisinage.

3.2.2 Identification des particules de surface

Notre méthode pour identifier les particules de surface consiste pour une particule à compter le nombre de cellules vides dans son voisinage et qui sont du même niveau qu'elle. Comme les particules situées à l'intérieur du fluide ont toutes leurs cellules voisines qui contiennent des particules, nous pouvons dire que les particules de surface sont les particules qui ont au moins une cellule vide dans leur voisinage (Figure 3.5). Il est évident que cette méthode n'est pas d'une précision optimale, elle est cependant assez précise pour déterminer si une particule est près de la surface ou non. S'il est nécessaire d'avoir une représentation plus précise de la surface du fluide, il est possible d'utiliser le critère complémentaire de la distance d'une particule par rapport aux objets.

Cette méthode ne fonctionne que si toutes les particules sont sur le même niveau de la grille, nous devons alors ajouter une nouvelle contrainte dans le cas de la simulation multi-résolution : une particule en contact avec une particule plus grande ne doit pas être considérée comme une particule de surface. En effet cela signifie qu'elle se situe à l'intérieur du fluide. Ce critère est très facile à évaluer puisque cette information est obtenue durant la phase de recherche de voisinage.

3.2.3 Identification des particules de bordure

Notre méthode d'identification des particules de bordure est similaire à la méthode utilisée pour les particules de surface mais nous devons examiner les cellules de la grille avec un plus grand rayon de recherche afin d'obtenir l'épaisseur désirée (Figure 3.5). En effet nous devons garder une bordure épaisse entre chaque niveau afin de garantir

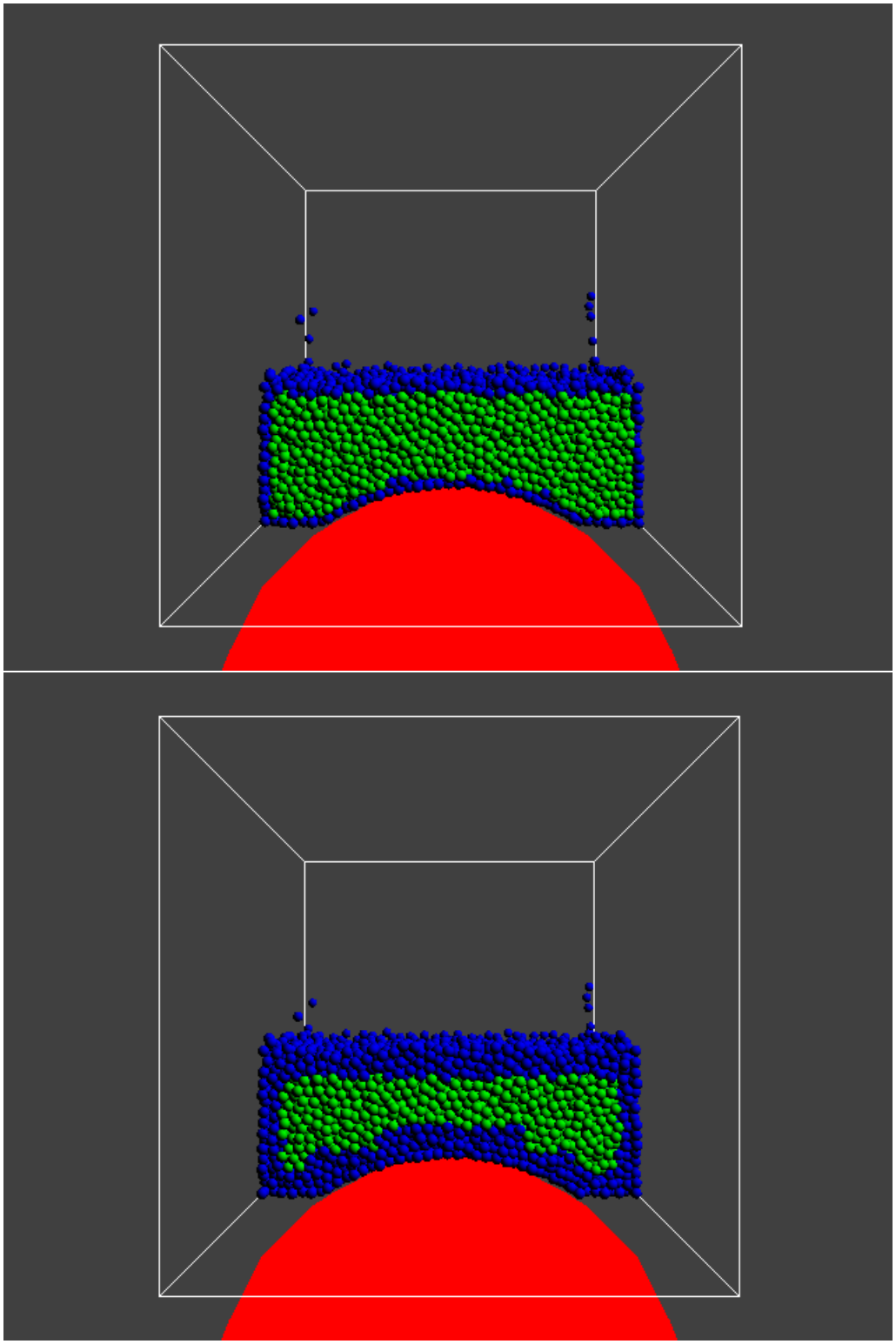


Figure 3.5 – L'image du haut représente les particules de surface en contact avec les objets ou à la surface du fluide. L'image du bas représente les particules de bordure qui incluent les particules de surface et représente une description plus large de la surface du fluide. Les critères basés sur l'utilisation de la grille laissent apparaître cette structure de grille (effet de « marches d'escalier »).

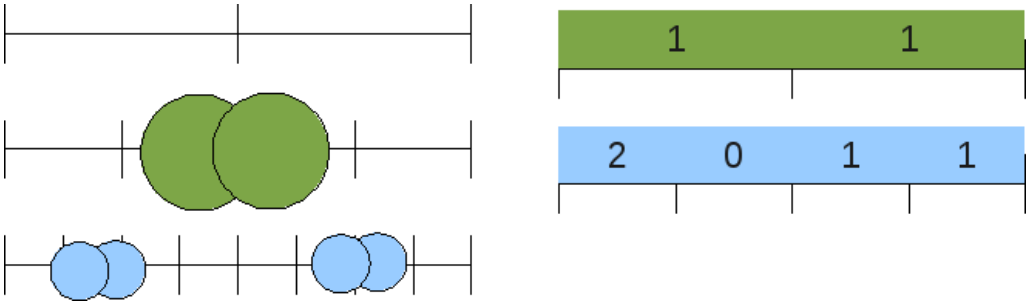


Figure 3.6 – Seconde grille hiérarchique dont les cellules contiennent le nombre de particules situées dans les cellules filles des niveaux inférieurs.

qu'aucune particule de niveau supérieur ne puisse atteindre la surface du fluide. Plutôt que de vérifier les cellules du niveau de la particule, nous utilisons les cellules du niveau immédiatement supérieur comme si la particule considérée était d'un niveau supérieur et possédait donc un rayon de recherche deux fois plus grand.

Cependant les niveaux supérieurs de la grille ne contiennent des informations que sur les particules de niveaux supérieurs, alors que pour détecter si une particule est une particule de bordure, nous devons avoir des informations sur les particules de même niveau situées autour de la particule considérée. Nous ne pouvons toutefois pas compter directement les particules situées dans le niveau de la particule puisque un rayon de recherche plus grand implique une baisse de performance, le problème étant similaire à celui existant pour la recherche de voisinage.

Pour résoudre ce problème, nous utilisons une deuxième grille hiérarchique dont les cellules contiennent des informations sur les particules situées dans les niveaux inférieurs. Cette information est ajoutée lors de l'insertion dans la grille hiérarchique, chaque particule incrémentant un compteur de particule dans sa cellule parente au niveau supérieur (Figure 3.6). Chaque particule ayant besoin d'information sur les particules plus petites situées dans son rayon de recherche peut alors obtenir le nombre de ces particules dans la deuxième grille hiérarchique en utilisant les mêmes indices de cellules calculés pour la phase de recherche de voisinage. Le nombre de particules de chaque niveau est alors pondéré en fonction de leur taille et additionné :

$$n = n_l + \frac{1}{4} n_{l-1} \tag{3.5}$$

où n_l est le nombre de particules du même niveau que la particule considérée et n_{l-1} est le nombre de particules du niveau inférieur. En utilisant cette deuxième grille il est alors possible d'estimer, pour une zone donnée, la densité de particules de deux niveaux adjacents de la grille.

L'Algorithme 3.3 présente une version modifiée de la méthode de recherche des particules candidates présenté dans l'Algorithme 3.1. Cette version de l'algorithme

Algorithme 3.3 Recherche des particules voisines.

```
set<particle*> getCandidateParticles(particle p) {
 // List of potential colliding particles
 set<particle*> candidates;
 point pos = p.getPosition();
 // Particle counter
 int pCount = 0;

 // Look for candidates in only two levels
 for each (int l = p.level; l <= p.level+1; l++) {
  index originCell = computeGridCell(pos,l);
  for(int z = -1; z <= 1; z++)
   for(int y = -1; y <= 1; y++)
    for(int x = -1; x <= 1; x++) {
     // Compute corresponding neighboring grid cell
     index gridCell = originCell + index(x, y, z);
     // Get smaller particle count
     pCount = getCountFromLowerLevel(gridCell, l);
     // Get all particles stored in this cell
     pCount += getCandidateFromCell(gridCell,
                                    candidates, l);
     if (pCount > 0)
      // Increment cell count for that level
      p.cellCount[l]++;
    }
 }
 return candidates;
}
```

prend en compte la deuxième grille utilisée pour estimer la densité de particules autour de la particule courante.

3.2.4 Élimination des particules isolées

Un des problèmes fréquent des méthode de simulation de fluide adaptative est que des petites particules peuvent se trouver isolées dans des zones contenant des particules plus grandes sans avoir assez de particules voisines de la même taille pour être capable de fusionner. Ce problème est assez gênant pour notre méthode de simulation puisque pour qu'une particule puisse fusionner vers un niveau plus élevé il est nécessaire qu'elle se situe dans une zone homogène du fluide, afin que ces nouvelles particules n'entrent pas en contact avec les particules les plus petites. Une solution serait de fusionner les petites particules avec des particules plus grandes situées à proximité, cependant nous aurions dû alors mettre en place un schéma de fusion /

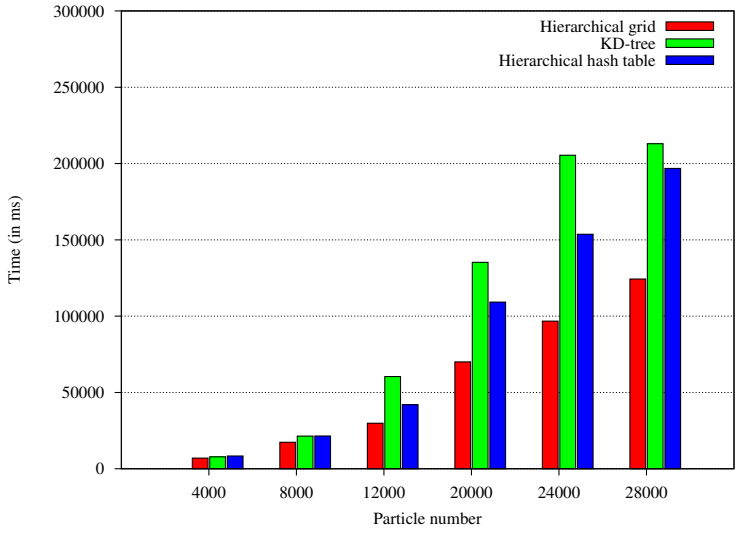


Figure 3.7 – Comparaison des performances de la phase de recherche de voisinage pour les trois structures.

subdivision beaucoup plus complexe pour gérer des ensembles composés de particules de tailles très différentes. Nous avons préféré une façon plus simple d'éliminer ces particules isolées : les particules qui sont détectée comme isolées sont stockées dans une liste, puis elles sont simplement utilisées pour fusionner avec d'autres particules isolées de même taille même si elles sont très éloignées les unes des autres. Cette opération introduit évidemment des erreurs dans la simulation mais qui ne sont pas remarquables puisque cette opération se produit sur des particules situées à l'intérieur du fluide. De plus, nous limitons le nombre de fusions lors de chaque itération pour ce type de particules.

La détection de ces particules est relativement directe puisque basée sur une modification des critères évalués lors de la recherche de voisinage. Ces particules sont en effet des particules qui ne sont pas des particules de bordure (donc situées à l'intérieur du fluide) et n'ayant aucune voisine de leur taille. Cette information est simplement calculée lors de la phase de recherche de voisinage.

3.3 Résultats

Nous avons testé les structures de grilles hiérarchiques et de tables de hachage hiérarchiques par rapport au KD-tree pour le problème de la recherche de voisinage comme le décrit la Section 3.1, sans utiliser de schéma adaptatif. La simulation est constituée de particules de trois tailles différentes qui sont réparties sous forme de couches. La proportion de petites particules est plus importante pour obtenir une simulation ayant des proportions aussi proches que possible d'une simulation adapta-

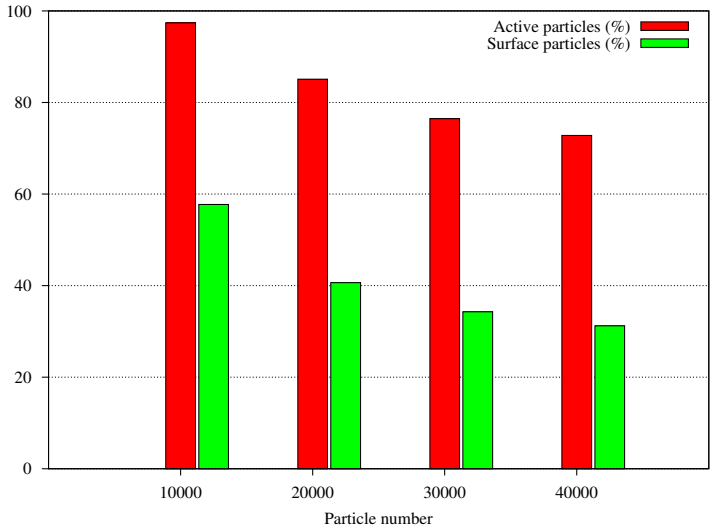


Figure 3.8 – Pourcentage de particules de surface et de particules actives lors de la simulation adaptative.

tive. Chaque structure est effacée à chaque pas de temps de la simulation puis remplie avec les nouvelles positions des particules. La Figure 3.7 montre le temps total (en ms) pour effectuer 500 itérations de la simulation. Les résultats montrent que la grille hiérarchique est la structure la plus performante des trois. La table de hachage hiérarchique, bien que moins performante, est intéressante si l'utilisation mémoire est un critère important (la table de hachage nécessite 117Ko contre 331Ko pour la grille pour une simulation de 40K particules) mais le surcoût des collisions de hachage est trop important lorsque le nombre de particules dans la simulation est faible. Enfin le KD-tree présente des résultats intéressants avec un faible nombre de particules mais le coût de la reconstruction de l'arbre devient trop important lorsque ce nombre augmente.

Nous avons aussi testé l'implémentation de la méthode adaptative décrite dans la Section 3.2 afin de simuler l'évolution d'un fluide composé de particules de différentes tailles. Nous avons effectué les tests sur des simulations avec un nombre croissant de particules durant 500 itérations de simulation adaptative. La Figure 3.8 montre le pourcentage de particules actives pendant toute la simulation, c'est-à-dire le nombre de particules présentes dans la simulation après les étapes de fusions et de subdivision. Cette figure montre aussi le pourcentage de particules détectée comme étant des particules de surface. Ces particules peuvent être directement utilisées pour effectuer l'extraction de la surface du fluide. Les résultats montrent que le schéma adaptatif est plus intéressant lorsque le nombre de particules augmente. En effet, dans un fluide contenant peu de particules le nombre de fusions est insuffisant pour compenser significativement le coût de l'évaluation des critères de fusion. Cependant, le mécanisme

de détection des particules de surface est intéressant pour n'importe quel nombre de particules puisque le problème de l'extraction de la surface du fluide est primordial même pour une simulation de fluide non-adaptative.

3.4 Implémentation et discussion

Nous avons expérimenté notre méthode de simulation adaptative sur GPU avec l'interface de programmation CUDA [1] qui permet de programmer avec le langage C sur les GPU NVIDIA. Nous avons remarqué que CUDA n'est pas adapté pour la simulation multi-résolution. En effet, bien que la structure de grille hiérarchique soit plus efficace que la grille uniforme pour cette tâche, la méthode adaptative est moins performante qu'une méthode classique. Nous avons trouvé plusieurs raisons à ces mauvaises performances :

- L'accès à la mémoire du GPU est très coûteux et les méthodes multi-résolution nécessitent de réaliser beaucoup plus d'accès à la mémoire que les méthodes classiques.
- L'évaluation des branchements dans un programme est une opération coûteuse puisque le GPU requiert que chaque *thread* exécute les mêmes instructions. Lorsque des particules de plusieurs tailles sont présentes dans la simulation elles exécutent généralement des instructions différentes en fonction de leur taille.
- L'évaluation des critères est aussi une opération très coûteuse. De plus la fusion et la subdivision de particules sont des opérations difficiles à réaliser avec CUDA.

Il est très probable que les évolutions futures des GPU permettront d'utiliser plus efficacement les méthodes adaptatives, la structure de grille hiérarchique permettra alors de traiter efficacement l'étape de recherche de voisinage.

La plupart des méthodes adaptatives (ainsi que la notre) utilisent uniquement des petites particules pour représenter la frontière du fluide. De cette manière la surface visible du fluide sera composée de petites particules mais aussi la surface invisible du fluide qui est en contact avec les objets de la scène. Cette configuration présente l'avantage de simuler avec beaucoup de précision la réaction du fluide par rapport à la géométrie de la scène, l'inconvénient est qu'il est nécessaire d'effectuer beaucoup de tests de collision entre ces particules et la géométrie même si les particules situées au centre du fluide ne sont pas considérées.

Lorsque le fluide est simulé dans une boîte, un critère fréquemment utilisé est d'adapter la taille des particules en fonction de leur distance par rapport au fond de la boite : plus une particule se situe profondément dans le fluide plus elle sera grande. Ce critère n'est utilisable que dans le cas d'une scène simple puisque les particules ne peuvent simuler avec précision les collisions avec des objets beaucoup plus petits

1. http ://www.nvidia.com/cuda

qu'eux. Nous pouvons quand même proposer une généralisation de ce critère : il est possible d'augmenter la taille d'une particule tant que les objets avec lesquels elle est en collision ne sont pas plus petits qu'elle. L'idée représentée par ce critère est que les particules doivent être petites là où les objets de la scène sont petits mais qu'il est possible d'approximer les collisions en augmentant la taille des particules si les objets sont plus grands. En effet, nous pouvons penser qu'une grande particule aura un comportement similaire à un groupe de particules plus petites face à des objets beaucoup plus grands qu'elles.

Afin de prendre en compte ce critère il serait possible d'utiliser une structure hiérarchique unique permettant de stocker des information sur les particules de plusieurs tailles différentes aussi bien que sur les objets composant la scène. De cette manière, une particule pourrait fusionner jusqu'à atteindre une taille du même ordre que la taille des objets situés à proximité.

Enfin, il serait possible de prendre en compte d'autres critères comme la distance des particules par rapport à l'observateur en maintenant une distribution de particules plus fine près de l'observateur. Il deviendrait cependant difficile d'effectuer des mouvement de caméra très rapides.

3.5 Conclusion

Nous avons décrit dans ce chapitre une structure de grille hiérarchique (ainsi que son extension en table de hachage hiérarchique) dans le cadre des méthodes de simulation de fluide lagrangiennes multi-résolution. Cette structure montre un avantage significatif pour la phase de recherche de voisinage par rapport aux structures de données traditionnellement utilisées pour cette tâche comme le KD-tree. Les performances sont particulièrement intéressantes pour la simulation d'un grand nombre de particules et elle est de plus très facile à implémenter et à utiliser puisqu'elle est une évolution des grilles uniformes.

Il est aussi possible d'utiliser cette structure pour résoudre différents problèmes comme l'évaluation des critères de fusion / subdivision des méthodes adaptatives. Ces critères sont évalués à un niveau local qui, bien que moins précis qu'un critère global, permet d'obtenir des résultats intéressants.

La perspective la plus intéressante de ces travaux est l'utilisation des structures hiérarchiques présentées dans les Chapitres 2 et 3 afin de produire une simulation adaptative dont le critère principal d'adaptation est la taille des objets situés à proximité. Le rendu du fluide étant une étape importante de la simulation nous essayons aussi d'utiliser la structure hiérarchique pour cette tâche dans le Chapitre suivant.

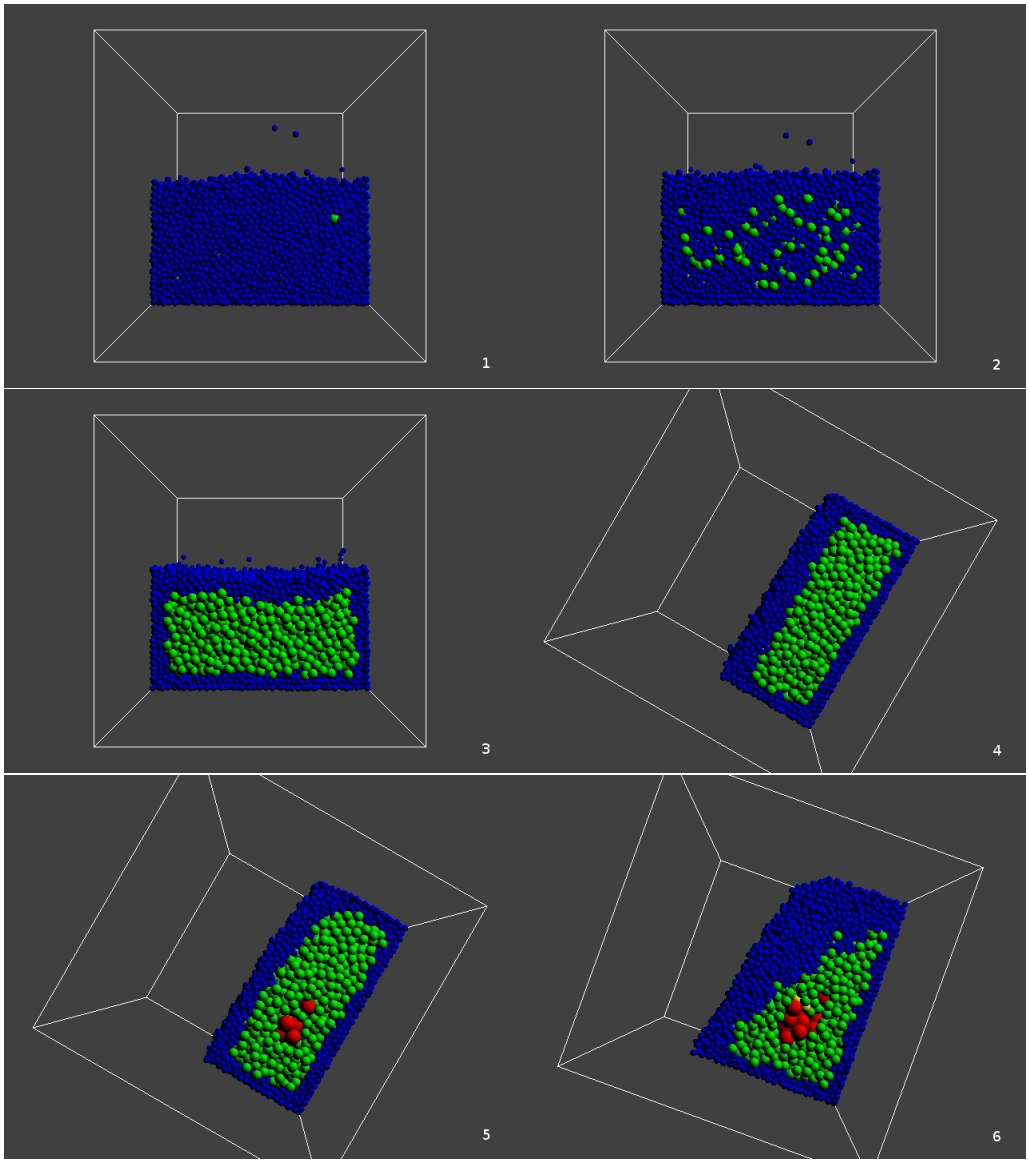


Figure 3.9 – Les premières images montrent le processus de fusion en action dont le résultat est une couche uniforme de particules de niveau 2 (image 3). La boîte est ensuite inclinée permettant l'apparition de particules de niveau 3 (image 5) puis la subdivision de particules de niveau 2 due à l'écoulement du fluide.

Chapitre 4

Visualisation de la surface du fluide

La problématique de la visualisation de la surface d'un fluide est de fournir une méthode permettant de représenter l'évolution de cette surface de manière harmonieuse, produisant des transitions douces entre chaque image tout en préservant les détails caractéristiques. Une telle méthode doit être capable de traiter les particules isolées ou en petits groupes et les particules formant des agglomérats. Cette méthode doit aussi permettre de traiter efficacement la fusion et la séparation de particules ou groupes de particules, et également être capable de bénéficier des structures de données hiérarchiques afin de proposer une implémentation efficace et éviter les calculs dans les zones dépourvues de fluide. Enfin, elle doit permettre de traiter des distributions de particules irrégulières et des distributions de particules adaptatives, avec des particules dont la taille peut changer au cours du temps. D'une manière générale les surfaces implicites semblent appropriées pour gérer ces contraintes, elles sont d'ailleurs utilisées par plusieurs auteurs dans le domaine de la visualisation de fluides. Les surfaces variationnelles en particulier présentent plusieurs avantages dans le contexte de la simulation de fluide :

- Il est possible de représenter une surface avec une distribution de particules irrégulière ou composée de particules de tailles différentes.
- Il est également possible d'obtenir une plus grande cohérence temporelle entre plusieurs images successives.

Les surfaces variationnelles n'ont pourtant à notre connaissance jamais été utilisées pour représenter la surface des fluides. Dans le cas des méthodes utilisant des particules comme éléments constitutifs du fluide, les méthodes représentant les particules sous forme de *blobs* sont les plus fréquentes. Cette popularité provient certainement de leur simplicité, elles ne sont cependant pas idéales pour cette utilisation.

Nous présentons dans ce chapitre nos expérimentations sur les surfaces implicites dans le but d'obtenir une surface du fluide de *qualité*, c'est à dire une surface qui ne présente pas de discontinuités, qui doit être assez proche des particules pour préserver les détails caractéristiques du fluide mais pas trop proche pour éviter de mettre en évidence les particules de la simulation. Nous avons pour cela utilisé des représentations à base de *blobs* et de surfaces variationnelles décrites dans la Section 1.4 de l'état de l'art.

4.1 Rendu à base de blobs

Les méthodes lagrangiennes utilisant des particules, une représentation de la surface sous forme de *blobs* semble être une méthode appropriée. De plus, la conversion des particules en *blobs* est directe : les *blobs* représentent des sphères ayant la capacité de s'agglomérer.

Nous avons dans un premier temps utilisé le moteur de lancer de rayons libre POV-Ray [1] qui permet de visualiser efficacement des scènes composées de *blobs* sans extraire de maillage. Plusieurs paramètres permettent d'influer sur le résultat final, en particulier : la force, le seuil et le rayon du *blob*.

- Le rayon ne détermine pas le rayon visible mais correspond plutôt à la zone d'interaction de chaque *blob*.
- Le seuil définit le niveau de l'isosurface qui est visualisée. Le rayon, en parcourant le champ défini par l'isosurface, s'arrête au point ayant une valeur identique à celle du seuil. Le seuil détermine donc la partie visible de chaque *blob*.
- La force est le paramètre spécifiant la force du champ au centre du *blob*. Une valeur positive indique que le *blob* attire les autres composants de la surface alors qu'une valeur négative les repousse. Plus la force est grande, plus la partie visible de chaque *blob* est importante. La valeur de la force et celle du seuil sont interdépendantes : multiplier la force par deux donne le même résultat que diviser le seuil par deux. Cependant la force peut être différente pour chaque *blob* alors que le seuil est commun à l'ensemble des *blobs* ce qui permet d'augmenter ou de diminuer la partie visible de chaque *blob*.

La valeur de l'isosurface en un point x de l'espace peut être calculée en utilisant la fonction suivante :

$$f(x) = F.\left(1 - \left(\frac{|x - x_0|}{r}\right)^2\right)^2 \tag{4.1}$$

où x_0 représente le centre d'un *blob*, r son rayon et F sa force.

1. http ://www.povray.org

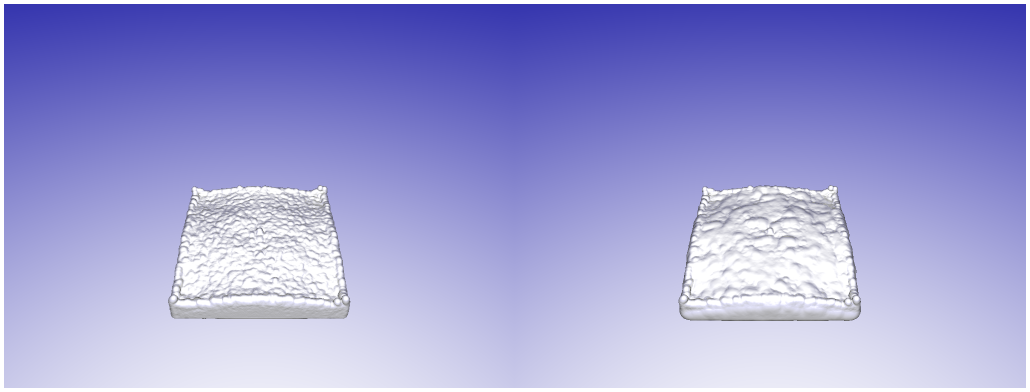


Figure 4.1 – La surface de l'image de gauche est calculée en utilisant uniquement les particules de surface. La surface de l'image de droite est calculée en utilisant toutes les particules, les particules situées à l'intérieur ayant un rayon plus grand que celles situées à la surface. Le résultat de l'utilisation des particules intérieures est une surface présentant moins d'irrégularités, mais visiblement déformée sur les bords.

Lors de la construction de la surface du fluide, l'approche naïve consiste à utiliser l'ensemble des particules de la simulation. Cependant, il est généralement possible de ne considérer que les particules présentes à la surface du fluide. En effet l'utilisation de la totalité des particules n'est pas nécessairement pertinente puisque les *blobs* situés à l'intérieur du fluide ne contribuent pas à la surface visible de l'extérieur. Nous avons donc d'abord effectué le rendu des particules de surfaces obtenues avec la structure hiérarchique (voir Section 3.2.2) en associant à chacune de ces particules un *blob* (image de gauche de la Figure 4.1).

L'utilisation des seules particules de surface permet de réduire significativement le temps de calcul de la scène. Cette utilisation présente cependant un inconvénient : il existe un espace vide à l'intérieur du fluide qui est alors visible par réfraction. Or il existe des cas où il est nécessaire de prendre en compte la totalité des particules, par exemple lorsque des objets sont visibles par transparence à l'intérieur du fluide ou lorsque les particules peuvent provenir de plusieurs fluides ayant des propriétés optiques différentes et se mélangeant. Dans un deuxième temps nous avons donc utilisé l'ensemble des particules de la simulation. Comme les particules situées loin de la surface devraient avoir une influence limitée sur la surface produite, nous avons utilisé pour ces particules des *blobs* de tailles plus grandes que pour celles de la surface. Le rayon des particules internes est ajusté de sorte que ces particules aient une influence sur la surface du fluide. Comme les particules utilisées ne sont présentes qu'à une certaine profondeur du fluide elles n'influeront que sur les grands volumes de fluide mais pas sur les particules isolées ou en petits groupes. Cela permet donc d'atténuer les irrégularités de la surface du fluide dans les zones où le fluide est profond tout en conservant les détails sur les petits ensembles de particules (image de droite

de la Figure 4.1).

Bien que l'utilisation de *blobs* pour représenter la surface du fluide soit simple, les surfaces obtenues avec cette méthode sont souvent associées au terme « blobby » qui décrit leur caractère bosselé. De plus, le résultat dépend fortement des paramètres utilisés et leur ajustement manuel pour obtenir la surface la moins irrégulière possible n'est pas une opération intuitive. De même, il est difficile de déterminer la valeur du rayon la plus adaptée aux particules situées à l'intérieur et un mauvais choix de cette valeur produit des artefacts visibles sur la surface du fluide. Plus le rayon est important, plus les irrégularités et trous de la surface disparaissent, plus les détails de la surface disparaissent également ne laissant qu'une représentation grossière de la surface comme l'illustre la Figure 4.2. Ces constatations mettent en évidence le besoin d'une méthode qui ne nécessite pas d'intervention de la part de l'utilisateur et qui produise une surface de plus grande qualité. Nous proposons donc dans la section suivante de décrire la surface du fluide grâce au modèle des surfaces variationnelles.

4.2 Utilisation des surfaces variationnelles

Les surfaces variationnelles sont généralement définies par des contraintes placées sur la surface et des contraintes placées soit à l'intérieur soit à l'extérieur de la surface. Dans notre cas nous cherchons à obtenir des points passant par la surface du fluide. Nous connaissons par contre les cellules contenant des particules et celles qui sont vides, nous connaissons aussi la position des particules situées à la surface du fluide. Nous devons alors définir des points de contraintes à partir de ces données afin d'obtenir une surface qui passe entre les contraintes intérieures et extérieures.

La surface définie avec cette méthode doit pouvoir répondre aux prérequis suivants :

1. Elle doit être sphérique pour une particule isolée.
2. Elle doit être plane quand les particules sont alignées sur un même plan (cas d'une fluide au repos).
3. Elle doit être plus lisse que la surface obtenue par le modèle des *blobs* décrit précédemment.

Nous devons alors trouver une méthode de placement des contraintes qui permette de traiter indifféremment les cas des particules isolées, alignées ou en groupe. Dans cette section nous avons effectué nos expérimentations en deux dimensions ; dans les figures illustrant ces expérimentations, les points noirs correspondent aux contraintes extérieures et les points rouges aux contraintes intérieures.

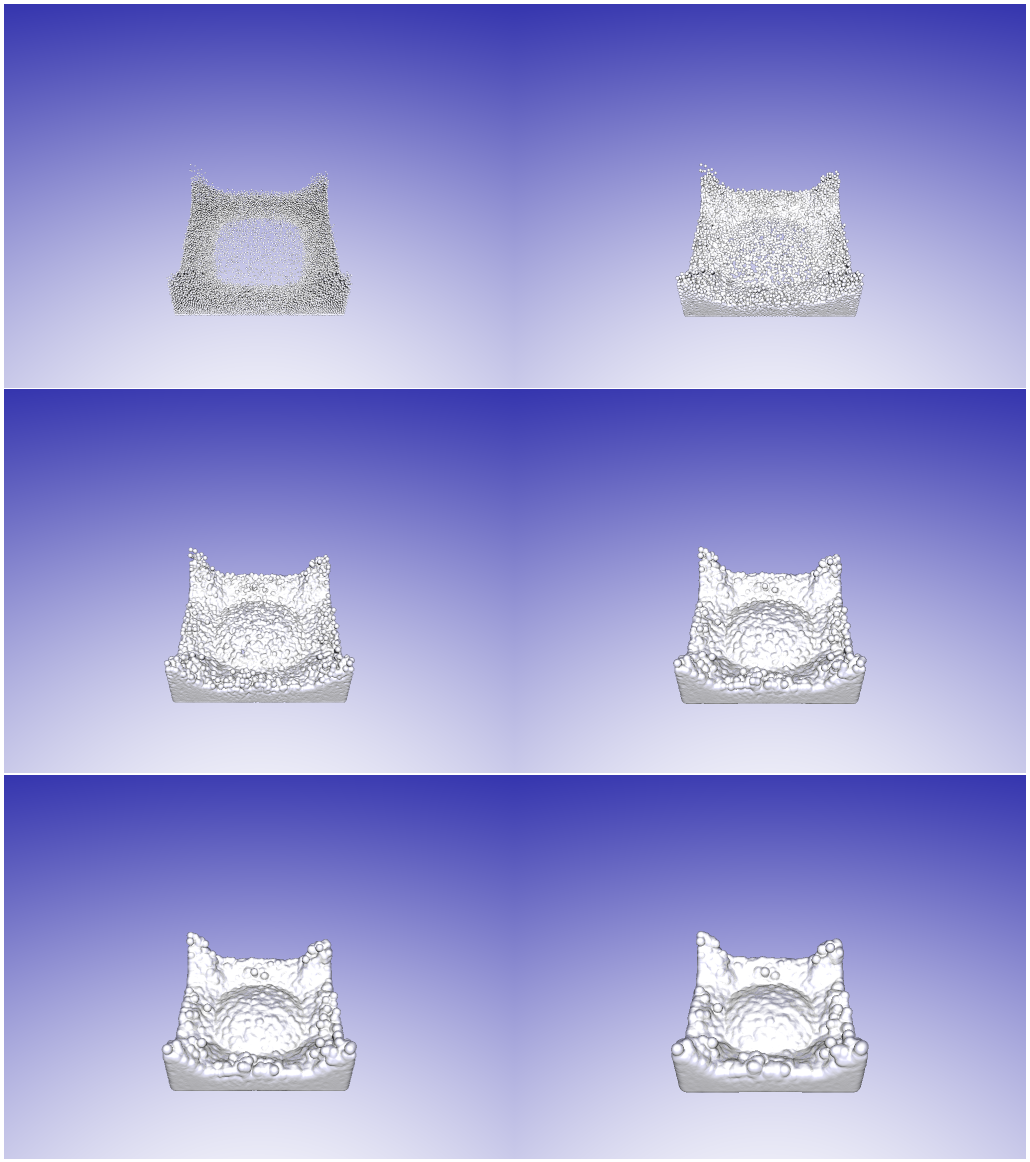


Figure 4.2 – Ces images montrent l'influence du rayon sur la surface produite. Le seuil a pour valeur 0.5, la force 2 et le rayon est multiplié par un coefficient allant de 1 à 6.

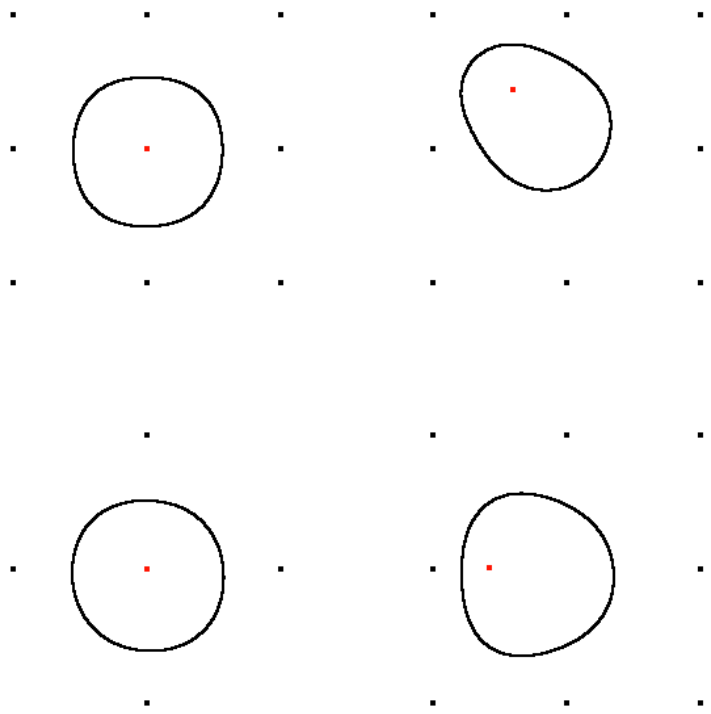


Figure 4.3 – Cette image représente des ensembles de contraintes et les surfaces qu'elles produisent. Les contraintes extérieures sont positionnés aux centre des cellules vides (fixes) et les contraintes intérieures aux centres des particules (mobiles). Lorsque la particules est située au centre de sa cellule la surface produite est circulaire (cas des images de gauche). Lorsque cette particule s'éloigne du centre de cette cellule, la surface est alors déformée (cas des images de droite). Le placement de la contrainte intérieure par rapport aux contraintes extérieures détermine donc la forme de la surface.

4.2.1 Placement des contraintes

Cas des particules isolées

Notre but est de reconstruire pour une particule isolée une surface de forme circulaire (sphérique en 3D), pour cela toutes les contraintes extérieures doivent être placées à égale distance de la contrainte intérieure puisque la surface produite se situe au barycentre des contraintes intérieures et extérieures. Ceci est vrai pour toutes les tailles de particules, ce qui signifie que les contraintes doivent être placées plus loin pour les particules de plus grande taille situées près de la surface. D'après nos essais quatre contraintes disposés de part et d'autre des axes x et y sont suffisantes pour représenter une surface de forme circulaire.

Les contraintes pourraient également être placées de façon à déformer les particules isolées en fonction de leur vélocité ou encore si elles sont en contact avec

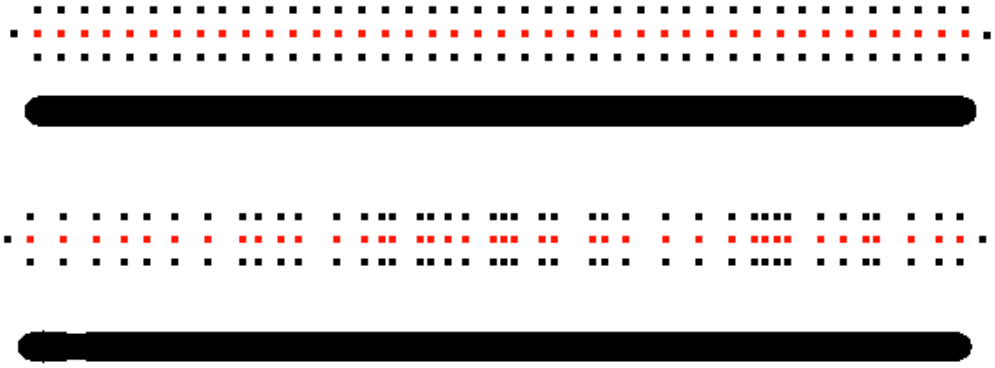


Figure 4.4 – L'image du haut représente les contraintes disposées de façon régulière permettant d'obtenir une surface plane. Dans cette image les contraintes intérieures sont placées sur une ligne horizontale et les contraintes extérieures sont situées à égale distance de ces contraintes sur une ligne verticale. L'image du bas représente les contraintes disposées de façon irrégulière sur la ligne horizontale, permettant d'obtenir une surface similaire.

un objet (une idée similaire est développée dans [YT10]), par exemple en aplatissant la surface après une collision, permettant ainsi d'éliminer l'effet de rigidité des simulations lagrangiennes composées d'un faible nombre de particules. Cette utilisation pourrait donner ainsi l'impression que la simulation sous-jacente est composée de beaucoup plus de particules que dans la réalité. La Figure 4.3 présente les résultats de nos différents essais sur les particules isolées.

Cas des particules alignées

Lorsque les particules sont alignées nous souhaitons construire une surface plane. Dans ce cas, les contraintes extérieures doivent être placées de part et d'autre des particules de surface, à égale distance des contraintes intérieures placées aux centres des particules. Dans cette configuration il est préférable que les particules soient positionnées à une distance régulière les unes des autres, cependant, une distribution irrégulière permet tout de même d'obtenir une surface plane (voir la Figure 4.4). Cette caractéristique est importante puisque les particules de la simulation sont rarement parfaitement alignées et il est nécessaire que cet aspect ne soit pas trop visible sur la surface du fluide.

Cas général

Dans un premier temps nous avons placé les contraintes intérieures au centre des particules et les contraintes extérieures au centre des cellules vides. Ce placement

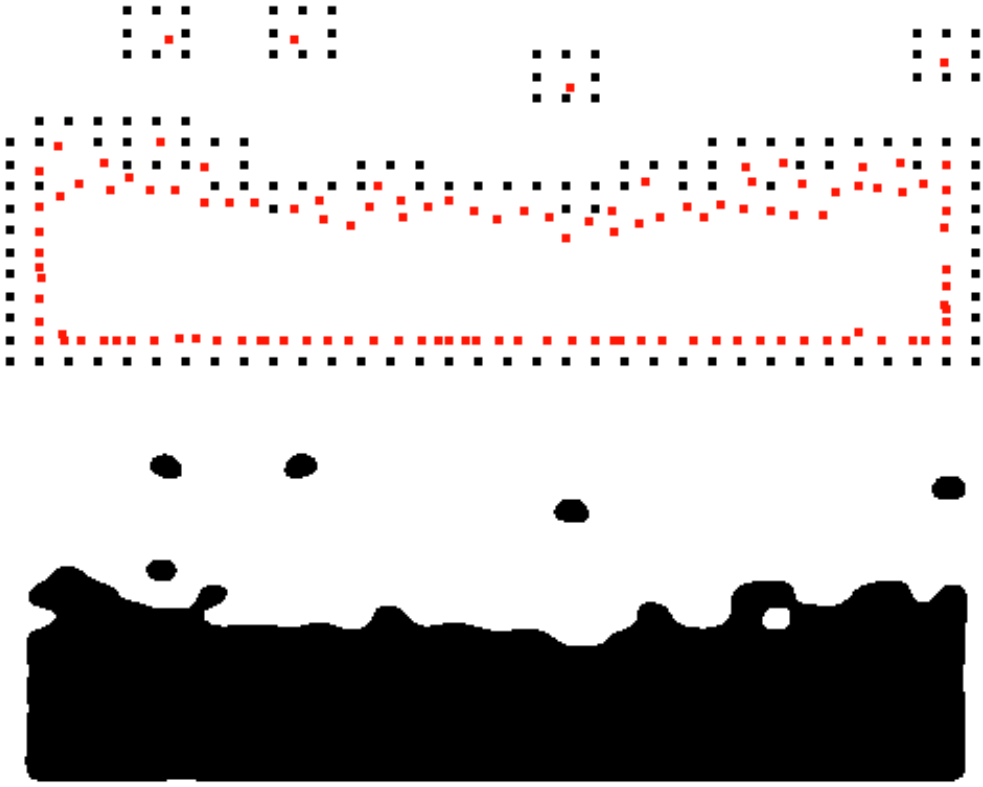


Figure 4.5 – Représentation d'une vue en coupe du fluide. Dans cette image les contraintes extérieures sont placées aux centres des cellules vides et les contraintes intérieures sont placées aux centres des particules de surface. Cette disposition est visible sur les particules isolées qui n'ont pas toutes la même forme.

n'est pas idéal car les particules peuvent se situer à la frontière entre deux cellules et la surface sera déformée comme le montre la Figure 4.3. Ceci implique que toutes les particules isolées auront un volume différent ce qui n'est pas souhaitable même si en pratique ces déformations seront peu visibles (Figure 4.5). Nous aurions pu utiliser toutes les particules de la simulation pour définir les contraintes intérieures de la surface. Cependant, les particules situées à l'intérieur du fluide n'apportent pas d'information sur la position de la surface du fluide. L'ajout de contraintes à l'intérieur du fluide est inutile et ralentit de plus la création de la surface variationnelle. Les contraintes intérieures que nous utilisons sont définies uniquement pour les particules de surface.

Le problème général du placement des contraintes pour les surfaces variationnelles n'a pas de solution évidente. Ce problème est par exemple évoqué dans les travaux de Reuter [Reu03]. Une autre façon de disposer les contraintes serait par exemple d'associer chaque particule de surface à une normale, cependant le placement des contraintes pour les particules isolées est problématique. De plus, lorsque deux parti-

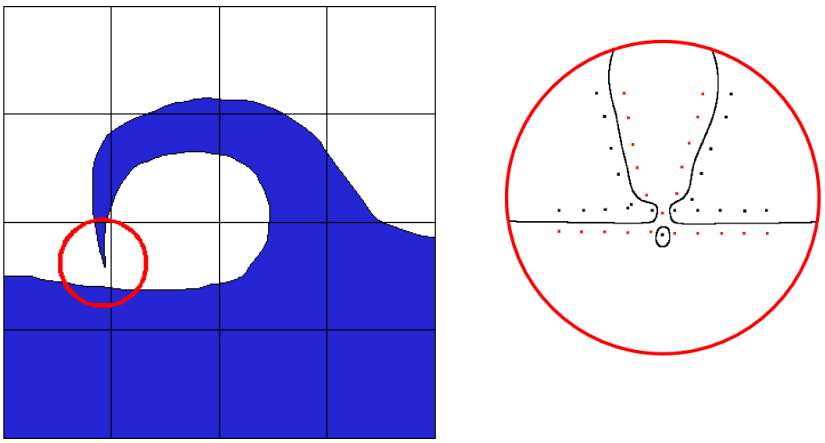


Figure 4.6 – L'image de gauche représente l'écoulement d'un fluide dans lequel la pointe de la vague rejoint le reste du fluide. Dans cette situation les normales des deux surfaces peuvent être en conflit. L'image de droite montre la déformation de la surface due aux contraintes de la pointe de la vague et l'apparition d'un espace vide à l'intérieur du fluide.

cules ou groupes de particules sont proches (Figure 4.6), il faut éviter que le placement des normales ne crée des conflits. En effet dans cette situation, les normales peuvent se croiser et se trouver trop près de la surface du fluide, ce qui peut créer des discontinuités.

Dans un deuxième temps, nous avons expérimenté une solution consistant à placer simplement une contrainte au centre de chacune des cellules de la grille utilisée pour la recherche de voisinage. La valeur d'une contrainte dépend de la distance entre le centre de la cellule et la particule la plus proche :

$$\begin{cases} -1, & si\, d \geq 2r \\ 0, & si\, d = r \\ +1, & si\, d = 0 \end{cases} \tag{4.2}$$

où d est la distance de la particule la plus proche et r son rayon. La valeur V de la contrainte associée à chaque cellule est donc interpolée en fonction de cette distance :

$$V = lerp\,(d/2r, -1, +1) \tag{4.3}$$

où $lerp$ est l'opération d'interpolation linéaire.

Ce schéma de placement des contraintes permet de prendre facilement en compte le cas des surfaces proches puisque les contraintes sont disposées à intervalles réguliers. Le nombre de contraintes utilisé est toutefois élevé puisqu'il dépend du nombre de cellules dans la grille. Des cellules trop grandes peuvent donc générer une surface trop approximative mais plus rapide à calculer. Au contraire, des cellules trop petites

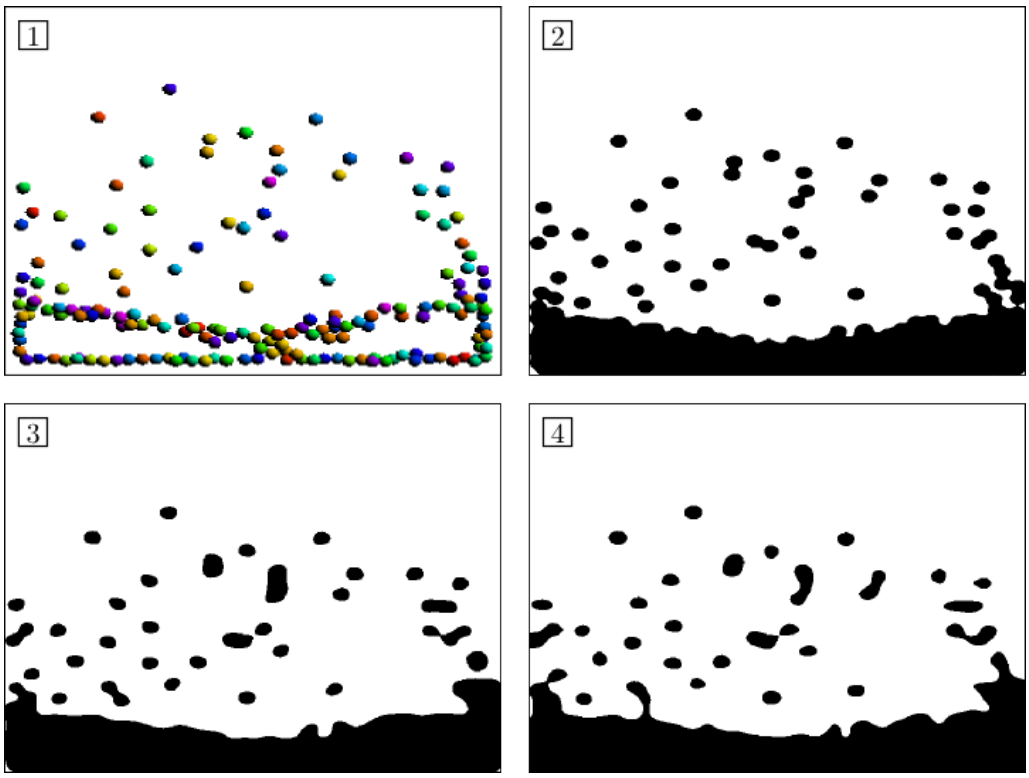


Figure 4.7 – Cette figure présente les surfaces produites à partir d'une tranche de particules de notre simulation de fluide (1) par la méthode des *blobs* (2), ainsi que nos deux méthodes (3,4) utilisant les surfaces variationnelles.

mettent en évidence les particules de la simulation tout en augmentant le temps de calcul. L'inconvénient de cette approche est une légère perte de précision, visible sur les particules isolées, due à la discrétisation des contraintes. Cette méthode est illustrée par la Figure 4.7 qui compare la surface produite par nos deux solutions utilisant les surfaces variationnelles ainsi que la méthode des *blobs*, cette dernière produisant nettement une surface beaucoup plus irrégulière.

L'utilisation d'une structure hiérarchique pourrait permettre de simplifier le placement des contraintes, en plaçant moins de contraintes dans les zones vides ou pleines du fluide tout en leur donnant des valeurs plus grandes que $+1$ ou -1 .

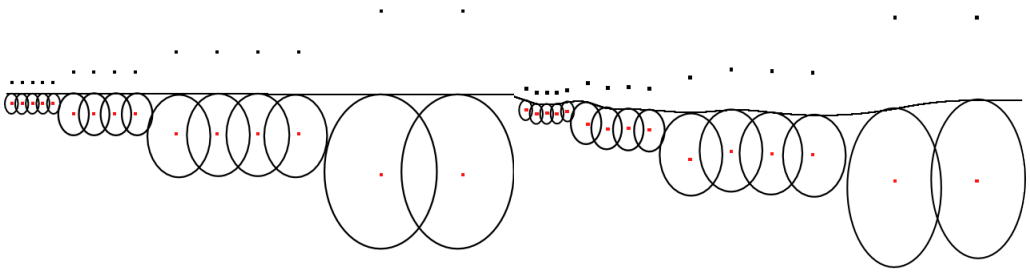


Figure 4.8 – Dans l'image de gauche les contraintes sont disposées afin qu'elles soient de plus en plus éloignées de la surface. La distance des contraintes intérieures par rapport à la surface est cependant toujours identique à celle des contraintes extérieures. Ce placement de contraintes simule des particules de plus en plus grandes situées près de la surface du fluide. La surface produite est régulière et ne présente pas de discontinuités. Au contraire si les particules sont disposées de manière irrégulière sur l'axe vertical, alors la surface produite est elle aussi irrégulière (image de droite).

4.2.2 Surfaces variationnelles et méthodes adaptatives

Les surfaces variationnelles permettent de reconstruire efficacement des surfaces planes avec peu de contraintes positionnées à des distances irrégulières les unes des autres comme le montre la Figure 4.4. Cette propriété est intéressante puisqu'elle pourrait permettre de reconstruire des surfaces planes composées de particules de taille différentes (Figure 4.8), de manière similaire à la méthode de Adams et al [APKG07]. Cette propriété permet de simuler un fluide avec de grandes particules dans les zones situées loin de l'observateur, tout en limitant l'aliassage lors de l'extraction de la surface.

4.2.3 Cohérence temporelle

Une des capacités intéressantes des surfaces variationnelles est la possibilité d'interpoler les surfaces de deux objets A et B, permettant de donner un effet de *morphing* dans lequel l'objet A se déforme progressivement pour se transformer en l'objet B. Cette possibilité peut être utilisée dans le cas de la simulation de fluide afin d'améliorer la cohérence temporelle des images produites, par exemple en calculant une image supplémentaire entre chacune des images produites initialement (Figure 4.9). Il serait ainsi possible de générer des transitions plus fluides entre chaque étape de la simulation, augmentant ainsi le réalisme de l'animation. Il pourrait aussi être utile

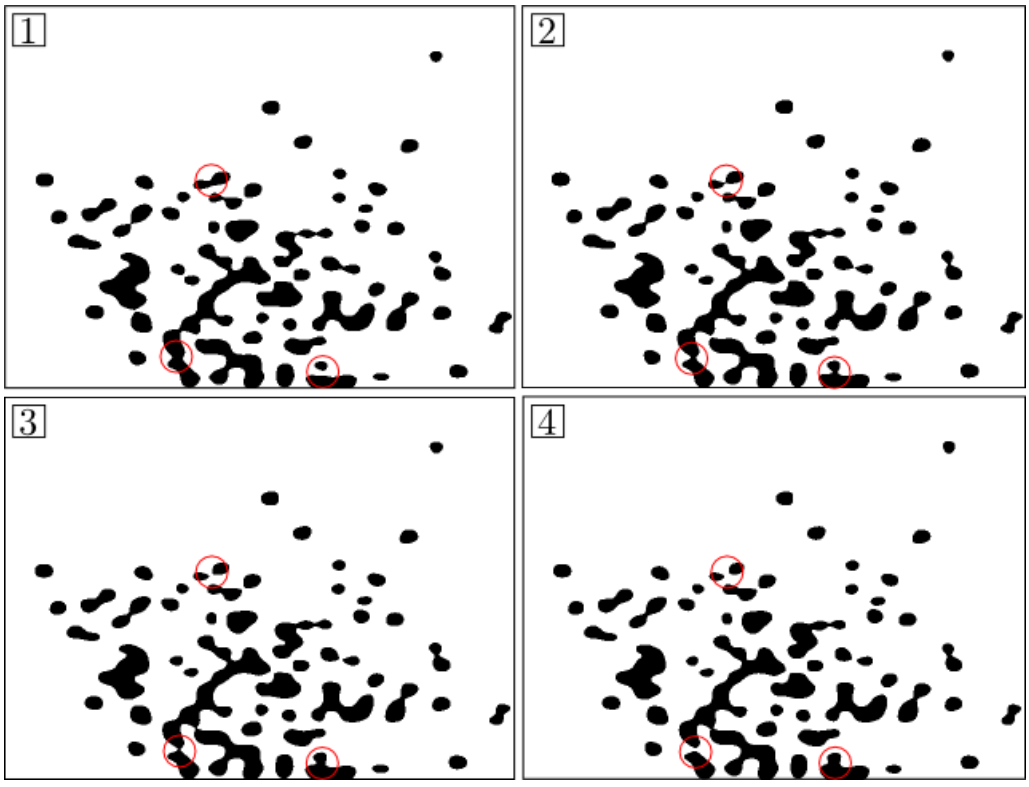


Figure 4.9 – Les images 1 et 4 représentent deux images de la simulation aux temps t et $t+2$. Les images 2 et 3 sont des images interpolées par rapport aux images d'origine. Les éléments entourés en rouge représentent les zones où l'interpolation produit un résultat satisfaisant.

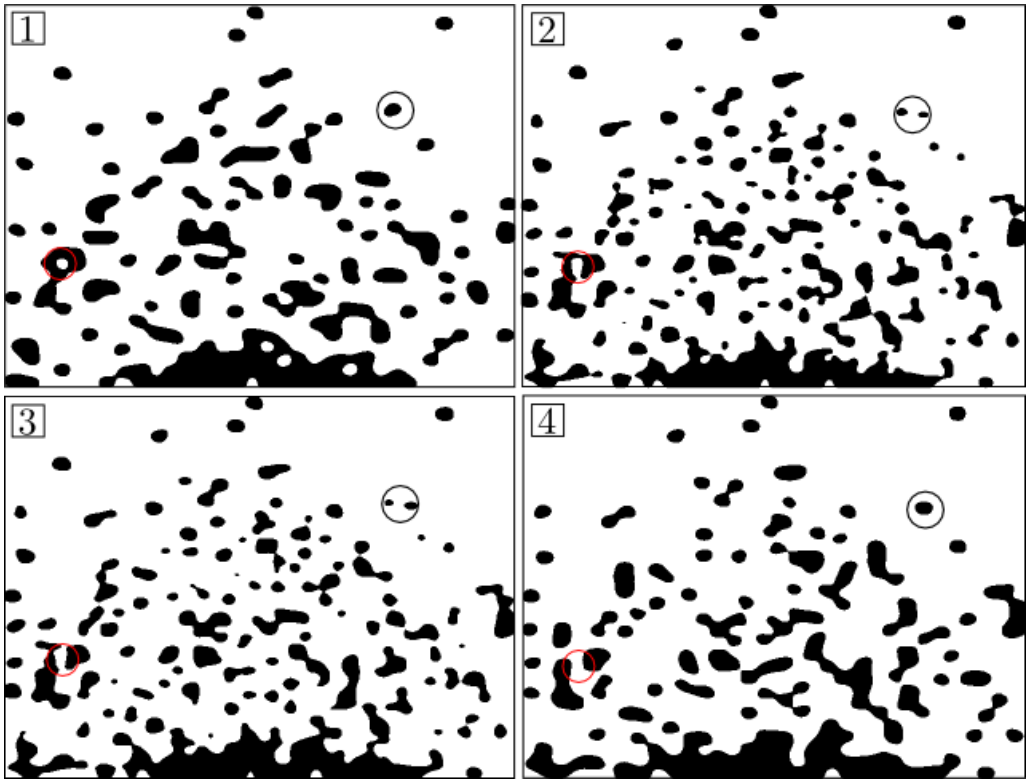


Figure 4.10 – Les images 1 et 4 représentent deux images de la simulation aux temps t et $t + 5$. La zone entourée en noir montre un résultat qui n'est pas satisfaisant puisque l'on observe un transfert de masse entre deux positions successives d'une même particule.

de ne calculer la surface du fluide que toutes les n images et de calculer les images intermédiaires par interpolation. De plus, cette capacité pourrait permettre de fournir un effet de flou de mouvement robuste avec les méthode lagrangiennes.

Il est cependant important que la différence entre les images interpolées ne soit pas trop importante. En effet, cette méthode ne tient pas compte du mouvement du fluide lui même, ce qui peut provoquer des incohérences dans l'interpolation (Figure 4.10). Pour obtenir des transitions plus correctes entre les images il faudrait donc prendre en compte la vélocité des particules en plaçant des contraintes supplémentaires dans la dimension temporelle.

4.3 Discussion et conclusion

La méthode des surfaces variationnelles est une méthode globale prenant en compte l'ensemble des contraintes décrivant la surface, ce qui est un des facteurs du coût de calcul important de cette méthode. Un des éléments responsables de ce coût est le système linéaire qu'il est nécessaire de résoudre. En effet, chacune des

contraintes est prise en compte dans la résolution du système linéaire de la surface, même les contraintes plus éloignées. Ce problème est encore plus important dans le cas de simulations en trois dimensions ou avec l'introduction de la dimension temporelle, ce qui réserve ce type d'utilisation aux applications non temps réel.

Une façon simple de diminuer ce coût de calcul est de subdiviser le domaine de résolution en plusieurs sous domaines, chacun produisant une surface variationnelle, par exemple en utilisant la méthode *multi-level partition of unity* [OBA+03]. Il est aussi envisageable d'utiliser les GPU qui sont bien adaptés à la résolution des systèmes linéaires.

La proximité des contraintes est aussi un problème pour cette méthode puisque des contraintes trop proches peuvent produire des résultats indéfinis. Ce problème apparaît dans les zones ayant une forte densité de contraintes, et peut être partiellement résolu en rééchantillonnant les contraintes.

Peu importe la méthode utilisée pour la construction de la surface implicite, il est généralement nécessaire de la convertir en une surface explicite plus simple à visualiser. L'extraction d'une surface implicite avec la méthode des *marching cubes* est une opération coûteuse puisqu'elle nécessite l'évaluation de la fonction implicite pour les points situés aux sommets de chaque cube. L'utilisation de la structure hiérarchique décrite dans le chapitre précédent est intéressante dans ce contexte car elle permet d'éliminer rapidement les zones de l'espace ne contenant pas de fluide. Ces zones vides ayant déjà été repérées lors de l'étape de recherche de voisinage, cette opération ne nécessite pas de calculs supplémentaires, et il est donc possible d'utiliser cette information sans avoir à calculer la valeur de la fonction implicite.

Les travaux sur les surfaces variationnelles présentés dans ce chapitre sont expérimentaux et seront développés par la suite. Cette méthode montre cependant un grand potentiel pour la représentation de la surface des fluides.

Conclusion

Nous avons vu dans ce document que l'utilisation de structures de données adaptées est un élément important dans les méthodes de simulation de fluides. Nous avons montré que les méthodes lagrangiennes en particulier nécessitent des structures accélératrices dans plusieurs phases de la simulation : la détection de collisions, la recherche de voisinage et la visualisation du fluide.

Résumé des contributions

La détection de collisions est un élément coûteux de la simulation. Le problème de cette étape est que les objets peuvent être répartis de manière irrégulière dans le domaine de simulation et peuvent être de tailles très différentes. Ils peuvent aussi se déplacer ou encore changer de taille au cours de la simulation. Dans ces situations les structures hiérarchiques sont nécessaires puisqu'elles s'adaptent facilement à des distributions irrégulières. Les structures arborescentes ne sont pas les plus efficaces à cause du coût de construction élevé qui empêche leur mise à jour à chaque pas de temps de la simulation. Les structures de hachage sont plus souvent employées pour cette tâche du fait de leur faible coût de construction et de leur faible utilisation mémoire. Ces méthodes souffrent cependant des collisions de hachage qui augmentent les tests de collisions. Nous avons donc proposé une structure de hachage hiérarchique qui permet de pallier ces problèmes. Premièrement notre structure utilise un schéma de hachage hiérarchique prenant en compte l'ensemble des niveaux présents dans la structure afin de bénéficier de la cohérence spatiale des objets. Ce schéma permet de réduire significativement les collisions de hachage et ainsi d'effectuer moins de tests de collisions inutiles. De plus notre structure permet de traiter rapidement les zones vides de la scène grâce à l'ajout d'une variable booléenne indiquant si les cellules des niveaux inférieurs contiennent des objets, comme pour une structure arborescente mais sans qu'il soit nécessaire de stocker un booléen pour chaque niveau.

La recherche de voisinage est une étape nécessaire des méthodes lagrangiennes puisque l'écoulement du fluide est déterminé par les interactions entre particules voisines. Cette étape est encore plus critique dans le cas des méthodes adaptatives qui

présentent des distributions de particules de tailles très différentes. Les structures accélératrices généralement employées pour cette étape comme les grilles uniformes ne sont pas adaptées dans ces situations puisqu'elles reposent sur l'utilisation de cellules de taille fixe et augmentent donc considérablement le coût de la recherche de voisinage pour les particules les plus grandes. Comme dans le cas de la détection de collisions, les grilles et tables de hachage hiérarchiques offrent une solution pertinente à ce problème en insérant chaque particule dans une cellule de taille appropriée permettant ensuite de réduire le nombre de cellules à visiter. La recherche de voisinage nécessite cependant une utilisation adaptée de ces structures. Nous avons donc proposé un schéma d'utilisation des grilles hiérarchiques consistant en une recherche ascendante des particules de tailles supérieures qui permet de déterminer efficacement les relations de voisinage entre des particules de tailles très différentes distribuées de manière aléatoire. Ce schéma peut de plus être amélioré lorsque les particules sont réparties en couches de particules de mêmes tailles. Nous avons ensuite utilisé la structure hiérarchique afin de déterminer les particules pouvant fusionner et celles devant se subdiviser en utilisant uniquement des informations locales obtenues dans la structure lors de la phase de recherche de voisinage. Cette structure nous a aussi permis d'obtenir simplement un ensemble de particules représentant la surface du fluide utilisable directement dans la phase de visualisation.

Cette étape de la visualisation du fluide est une des plus importantes pour le réalisme d'une simulation. Dans le cas des méthodes de simulation lagrangiennes, les méthodes de visualisation les plus couramment utilisées consistent à convertir les particules de la simulation sous forme de *blobs*. Cette représentation très simple produit néanmoins une surface très irrégulière et très bosselée mettant en évidence les particules de la simulation et qui n'est donc pas très convaincante. Nous avons proposé de représenter la surface du fluide avec la méthode des surfaces variationnelles qui possède un fort potentiel pour cette tâche puisqu'elle est capable de traiter efficacement des distributions de particules irrégulières. Nos premiers essais montrent que cette méthode permet de produire une surface de plus grande qualité que la méthode des *blobs*. Elle est de plus capable de prendre en compte la dimension temporelle afin de produire des transitions plus douces entre chaque image de la simulation.

Dans l'ensemble des travaux présentés nous avons utilisé la méthode DEM pour effectuer la simulation de fluide, ils sont évidemment utilisables avec n'importe quelle méthode représentant le fluide avec des particules.

Travaux futurs

Bien que les méthodes adaptatives permettent de diminuer le nombre global de particules dans la simulation, elles ne permettent généralement pas de diminuer le

nombre de tests de collisions. En effet, les particules en contact avec les objets de la scène sont souvent de la plus petite taille afin de calculer avec précision la réaction du fluide. Nous pensons qu'une telle précision n'est pas nécessaire dans tous les cas. En particulier, si la taille de l'objet (triangle) est très supérieure à celle de la particule, nous supposons que deux particules entrant en collision avec cet objet à la même vélocité ont la même réaction. Nous pensons donc qu'il est possible de fusionner les particules qui sont en collision avec des objets tant que ces objets sont plus grands que les particules, sans grande perte de précision. Au contraire les particules devraient se subdiviser en approchant d'objets plus petits qu'elles afin de ne pas simplifier leur réaction. Pour cela nous envisageons d'utiliser les structures hiérarchiques que nous avons proposées dans ce document. Une particule pourrait ainsi déterminer facilement son comportement en fonction de la taille des objets qui lui sont proches.

La méthode des surfaces variationnelles est très coûteuse en temps de calcul et une utilisation des structures accélératrices, en particulier des structures hiérarchiques nous semble appropriée. Par exemple, subdiviser le domaine de résolution en plusieurs sous-domaines associés à des cellules de la grille hiérarchique permettrait de simplifier le calcul de la surface. De même, la méthode des *marching cubes* pourrait également bénéficier de structures de données hiérarchiques. En effet l'évaluation de la fonction implicite de la surface aux sommets de chaque cube est coûteuse et inutile puisque la valeur de cette fonction est déjà connue dans les zones ne contenant pas de particules. Pour cette utilisation une structure hiérarchique permet de connaître la distribution spatiale des particules sur plusieurs niveaux afin d'éliminer rapidement les zones qui ne contiennent pas la surface du fluide. De plus, il n'est pas nécessaire de produire des triangles de la plus petite taille possible. Dans les cas où des particules de différentes tailles se situent près de la surface il serait alors possible de créer des triangles dont les tailles seraient adaptées à celles des particules. Cette utilisation permettrait alors, en plaçant de grandes particules près de la surface loin de l'observateur, de limiter l'aliassage souvent visible dans ces zones.

Bien que les structures hiérarchiques que nous avons présentées dans ce document permettent d'améliorer les différentes phases d'une simulation de fluide elles ne permettent pas cependant de simuler un fluide dans les applications temps réelles comme les jeux vidéo. Pour ce type d'applications nous imaginons une approche utilisée par exemple, pour la simulation de fumée dans les travaux de Kim et al. [KTJG08], consistant à créer une simulation basse résolution à laquelle seraient ajoutés des détails afin qu'elle ressemble à une simulation haute résolution.

Bibliographie

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. In *SIGGRAPH '07 : ACM SIGGRAPH 2007 papers*, page 48, New York, NY, USA, 2007. ACM.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9) :509–517, 1975.

[Bli82] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3) :235–256, 1982.

[BMF07] Robert Bridson and Matthias Müller-Fischer. Fluid simulation : Siggraph 2007 course notes. In *SIGGRAPH '07 : ACM SIGGRAPH 2007 courses*, pages 1–81. ACM, 2007.

[BR86] Jeremiah U Brackbill and Hams M Ruppel. Flip : A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.*, 65(2) :314–343, 1986.

[BT07] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *SCA '07 : Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217, 2007.

[BW97] Jules Bloomenthal and Brian Wyvill, editors. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997.

[cHcWyCO05] Chien chang Ho, Fu che Wu, Bing yu Chen, and Ming Ouhyoung. Cubical marching squares : Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum*, 24 :537–545, 2005.

[CMVHT02] Mark Carlson, Peter J. Mucha, R. Brooks Van Horn, III, and Greg Turk. Melting and flowing. In *SCA '02 : Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 167–174. ACM, 2002.

[CTM02] Hamish Carr, Thomas Theußl, and Torsten Möller. Modified marching octahedra for optimal regular meshes. In *SIGGRAPH '02 : ACM SIG-*

GRAPH 2002 conference abstracts and applications, pages 219–219. ACM, 2002.

[DC99] Mathieu Desbrun and Marie-Paule Cani. Space-time adaptive simulation of highly deformable substances. Rapport de recherche 3829, INRIA, December 1999.

[EL07] Mathias Eitz and Gu Lixu. Hierarchical spatial hashing for real-time collision detection. In *SMI '07 : Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, pages 61–70, Washington, DC, USA, 2007. IEEE Computer Society.

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph.*, 21(3) :736–744, 2002.

[FKN80] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. *SIGGRAPH Comput. Graph.*, 14(3) :124–133, 1980.

[FM96] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5) :471–483, 1996.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 2001.

[GG98] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2) :170–231, 1998.

[GHD03] Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. Simulating fluid-solid interaction. In *Graphics Interface*, pages 31–38, 2003.

[Har63] Francis H. Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. *Experimental arithmetic, high-speed computations and mathematics*, 1963.

[HCM06] Kyle Hegeman, Nathan A. Carr, and Gavin S. P. Miller. Particle-based fluid simulation on the gpu. In Vassil N. Alexandrov, Dick G. van Albada, Peter, and Jack Dongarra, editors, *International Conference on Computational Science (4)*, volume 3994 of *Lecture Notes in Computer Science*, pages 228–235, 2006.

[HHK08] Woosuck Hong, Donald H. House, and John Keyser. Adaptive particles for incompressible fluid simulation. *Vis. Comput.*, 24(7) :535–543, 2008.

[HHK09] Woosuck Hong, Donald H. House, and John Keyser. An adaptive sampling approach to particle-based fluid. In *Proceedings of Theory and Practice of Computer Graphics*, pages 69–76, 2009.

[HKK07a] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Construction of non-blobby surface from particle. In *Eurographics Short Paper*, pages 41–44, 2007.

[HKK07b] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Sliced data structure for particle-based simulations on gpus. In *GRAPHITE '07 : Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 55–62, New York, NY, USA, 2007. ACM.

[HKK07c] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics on gpus. pages 63–70, 2007.

[HNB+06] Ben Houston, Michael B. Nielsen, Christopher Batty, Ola Nilsson, and Ken Museth. Hierarchical rle level set : A compact and versatile deformable surface representation. *ACM Trans. Graph.*, 25(1) :151–175, 2006.

[HW65] Francis H. Harlow and Eddie J. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12) :2182–2189, 1965.

[IGLF06] Geoffrey Irving, Eran Guendelman, Frank Losasso, and Ronald Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *SIGGRAPH '06 : ACM SIGGRAPH 2006 Papers*, pages 805–811. ACM, 2006.

[Jen01] Henrik W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, Ltd., July 2001.

[Kel06] Micky Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Technical report, Department of Computer Science, University of Cophenhagen, January 9 2006.

[KTJG08] Theodore Kim, Nils Thürey, Doug James, and Markus Gross. Wavelet turbulence for fluid simulation. *ACM Trans. Graph.*, 27(3) :1–6, 2008.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4) :163–169, 1987.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04 : ACM SIGGRAPH 2004 Papers*, pages 457–462, New York, NY, USA, 2004. ACM.

[LLL06] M. B. Liu, G. R. Liu, and K. Y. Lam. Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. *Shock Waves*, 15 :21–29, March 2006.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Mel00] Stan Melax. Dynamic plane shifting bsp traversal. In *Graphics Interface*, pages 213–220, 2000.

[Mis03] B.K. Mishra. A review of computer simulation of tumbling mills by the discrete element method part ii-practical applications. *International Journal of Mineral Processing*, 71(1-4) :95–112, 2003.

[Mon92] J.J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1) :543–574, 1992.

[MW97] Heinrich Müller and Michael Wehle. Visualization of implicit surfaces using adaptive tetrahedrizations. In *DAGSTUHL '97 : Proceedings of the Conference on Scientific Visualization*, page 243. IEEE Computer Society, 1997.

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3) :463–470, 2003.

[OF02] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[OVSM98] J. Michael Owen, Jens V. Villumsen, Paul R. Shapiro, and Hugo Martel. Adaptive smoothed particle hydrodynamics : Methodology. ii. *The Astrophysical Journal Supplement Series*, 116(2) :155, 1998.

[PACT09] Mickael Pouchol, Alexandre Ahmad, Benoit Crespin, and Olivier Terraz. A hierarchical hashing scheme for nearest neighbor search and broad-phase collision detection. *journal of graphics, gpu, and game tools*, 14(2) :45–59, 2009.

[PCG10] Mickael Pouchol, Benoit Crespin, and Djamchid Ghazanfarpour. Hierchical structure for multi-resolution particle-based methods. In *Computer Animation and Social Agents (Short Papers Session)*, June 2010.

[PTB+03] Simon Premoze, Tolga Tasdizen, James Bigler, Aaron E. Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Comput. Graph. Forum*, 22(3) :401–410, 2003.

[Reu03] Patrick Reuter. *Reconstruction and Rendering of Implicit Surfaces from Large Unorganized Point Sets*. PhD thesis, Université Bordeaux 1, décembre 2003.

[Rho92] Clifford E. Rhoades. A fast algorithm for calculating particle interactions in smooth particle hydrodynamic simulations. *Computer Physics Communications*, 70(3) :478–482, 1992.

[RSH00] Erik Reinhard, Brian E. Smits, and Chuck Hansen. Dynamic acceleration structures for interactive ray tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 299–306, London, UK, 2000. Springer-Verlag.

[SBH09] Funshing Sin, Adam W. Bargteil, and Jessica K. Hodgins. A point-based method for animating incompressible flow. In *SCA '09 : Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 247–255, New York, NY, USA, 2009. ACM.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH '99 : Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[SY02] Lin Shi and Yizhou Yu. Visual smoke simulation with adaptive octree refinement. Technical report, Champaign, IL, USA, 2002.

[SZP07] Barbara Solenthaler, Yanci Zhang, and Renato Pajarola. Efficient refinement of dynamic point data. In *Eurographics Symposium on Point-Based Graphics*, 2007.

[TDOY01] Greg Turk, Huong Quynh Dinh, James F. O'Brien, and Gary Yngve. Implicit surfaces that interpolate. In *SMI '01 : Proceedings of the International Conference on Shape Modeling & Applications*, page 62. IEEE Computer Society, 2001.

[THM+03] Matthias Teschner, Bruno Heidelberger, Matthias Mueller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vision, Modeling, Visualization (VMV 2003)*, pages 47–54, 2003.

[TI08] Steven G. Parker Thiago Ize, Ingo Wald. Ray tracing with the bsp tree. pages 159–166, 2008.

[TO99] Greg Turk and James F. O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1999.

[Tsa02] Yen-hsi Richard Tsai. Rapid and accurate computation of the distance function using grids. *J. Comput. Phys.*, 178(1) :175–195, 2002.

[vKvdBT07] Kees van Kooten, Gino van den Bergen, and Alex Telea. Point-based visualization of metaballs on a gpu. In *GPU Gems 3*, pages 123–148. Addison-Wesley, 2007.

[WIK+06] Ingo Wald, Thiago Ize, Andrew Kensler, Aaron Knoll, and Steven G. Parker. Ray tracing animated scenes using coherent grid traversal. *ACM Trans. Graph.*, 25(3) :485–493, 2006.

[YT10] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *SCA '10 : Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2010.

[YWH+09] He Yan, Zhangye Wang, Jian He, Xi Chen, Changbo Wang, and Qunsheng Peng. Real-time fluid simulation with adaptive sph. *Comput. Animat. Virtual Worlds*, 20(2-3) :417–426, 2009.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. In *SIGGRAPH '05 : ACM SIGGRAPH 2005 Papers*, pages 965–972, New York, NY, USA, 2005. ACM.

[Zha04] Hongkai Zhao. A fast sweeping method for eikonal equations. In *Mathematics of Computation*, volume 74, pages 603–627, 2004.

[ZPvBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01 : ACM SIGGRAPH 2001 papers*, pages 29–36. ACM, 2001.

[ZSP08] Yanci Zhang, Barbara Solenthaler, and Renato Pajarola. Adaptive sampling and rendering of fluids on the gpu. In *Volume and Point-Based Graphics 2008*, pages 137–146, August 2008.

Résumé

La simulation de l'écoulement des fluides comme l'eau ou la fumée est un des phénomènes naturels les plus traités par la communauté de l'informatique graphique. Parmi ces méthodes, les méthodes lagrangiennes, représentant le fluide à l'aide de particules, sont souvent préférées car elles permettent d'utiliser moins de ressources, elles restent cependant coûteuses en temps de calculs et nécessitent l'usage de structures de données adaptées lors de différentes phases. Nous proposons donc d'utiliser une structure de hachage hiérarchique afin de traiter la phase de détection de collisions avec des objets répartis de manière irrégulière et de tailles très différentes. L'utilisation de plusieurs niveaux de hachages permet de bénéficier de la cohérence spatiale des objets et de réduire significativement les collisions de hachage. Les méthodes adaptatives permettent d'allouer efficacement les ressources de calculs dans les zones importantes du fluide en faisant varier la taille des particules de la simulation. Dans cette situation, une structure de hachage ou de grille hiérarchique permet de traiter efficacement la phase de recherche de voisinage en insérant chaque particule dans le niveau de la grille approprié mais aussi de déterminer les zones de fusion ou de subdivision grâce à des critères locaux. Enfin la phase de visualisation du fluide représente un défi car le résultat est directement sanctionné par le spectateur. Dans le cas des méthodes lagrangiennes, la méthode des blobs traditionnellement utilisée ne permet pas d'obtenir une surface de qualité. Nous utilisons pour cette tâche la méthode des surfaces variationnelles.

Mots-clés : simulation de fluides, structures hiérarchiques, méthodes adaptatives, visualisation, surfaces implicites.

Abstract

Fluid flow simulation such as water or smoke is one of the most covered natural phenomena in the computer graphics community. Among these methods, lagrangian ones which divide the fluid into a set of particles, are usually favored for their efficiency, however they require suitable data structures in several steps to reduce their high computation times. We suggest to use a hierarchical hash table to handle the collision detection step with irregularly distributed and objects with variable sizes. The use of several hash levels allows to benefit from objects spatial coherence and to drastically reduce hash collisions. Adaptive methods allow to efficiently allocate computational resources to significant fluid regions by varying particles sizes during the simulation. In this case a hierarchical grid or hash table structure allows to efficiently handle nearest neighbors search by inserting each particle in the appropriate grid level, and also allows to define appropriate merging or splitting regions with local criteria. Finally the visualization step is challenging because the end-user will appreciate the final result directly. For lagrangian methods, the most common way to deal with this problem is to use blob-based methods that do not yield satisfying results ; we use the variational surface method for this task.

Keywords : fluid simulation, hierarchical structures, adaptive methods, visualisation, implicit surfaces.